TUGAS AKHIR – KI141502

PERANCANGAN DAN IMPLEMENTASI PEMILIHAN CLUSTER HEAD DAN KOORDINASI ANTAR NODE SENSOR PADA JARINGAN SENSOR NIRKABEL DENGAN MODUL WIRELESS NRF24L01

MOCH IMAM ZARQONI
NRP 5113100154

Dosen Pembimbing I
Waskitho Wibisono, S.Kom., M.Eng., Ph.D

Dosen Pembimbing II
Hudan Studiawan, S.Kom., M.Kom.

Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI141502

PERANCANGAN DAN IMPLEMENTASI PEMILIHAN CLUSTER HEAD DAN KOORDINASI ANTAR NODE SENSOR PADA JARINGAN SENSOR NIRKABEL DENGAN MODUL WIRELESS NRF24L01

MOCH IMAM ZARQONI
NRP 5113100154

Dosen Pembimbing I
Waskitho Wibisono, S.Kom., M.Eng., Ph.D.

Dosen Pembimbing II
Hudan Studiawan, S.Kom., M.Kom.

Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESES – KI141502

DESIGN AND IMPLEMENTATION OF CLUSTER HEAD SELECTION AND INTER NODE COORDINATION FOR WIRELESS SENSOR NETWORK USING NRF24L01 WIRELESS MODULE

MOCH IMAM ZARQONI
NRP 5113100154

First Advisor
Waskitho Wibisono, S.Kom., M.Eng., Ph.D.

Second Advisor
Hudan Studiawan, S.Kom., M.Kom.

Department of Informatics
Faculty of Information Technology
Sepuluh Nopember Institute of Technology
Surabava 2017

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

PERANCANGAN DAN IMPLEMENTASI PEMILIHAN *CLUSTER HEAD* DAN KOORDINASI ANTAR SENSOR NODE PADA JARINGAN SENSOR NIRKABEL DENGAN MODUL WIRELESS NRF24L01

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
MOCH IMAM ZARQONI
NRP: 5113100154

Disetujui oleh Pembimbing Tugas Akhir:

1. Waskitho Wibisono, S.Kom., M.Eng., Ph.D. .....
   (NIP. 197410222000031001) (Pembimbing 1)

2. Hudan Studiawan, S.Kom., M.Kom. .....
   (NIP. 198705112012121003) (Pembimbing 2)

SURABAYA

JUNI, 2017

(Halaman ini sengaja dikosongkan)

PERANCANGAN DAN IMPLEMENTASI PEMILIHAN *CLUSTER HEAD* DAN KOORDINASI ANTAR SENSOR NODE PADA JARINGAN SENSOR NIRKABEL DENGAN MODUL WIRELESS NRF24L01

Nama Mahasiswa	**: MOCH IMAM ZARQONI**
NRP	**: 5113100154**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**
Dosen Pembimbing 2	**: Hudan Studiawan, S.Kom., M.Kom.**

Abstrak

Wireless Sensor Network (WSN) adalah sebuah jaringan yang terdiri dari node-node yang didistribusikan baik secara strategis mauun secara random yang bertujuan untuk mengamati suatu kejadian tertentu. Jaringan dalam tugas akhir ini dibangun menggunakan modul wireless nRF24L01+. Modul ini didesain untuk jaringan nirkabel yang membutuhkan daya sangat rendah. Sehingga cocok digunakan pada node-node dalam jaringan sensor nirkabel.

Jaringan sensor dalam tugas akhir ini dibuat dengan harapan semua node dapat mengirimkan data ke sebuah server / coordinator. Namun, apabila semua node mengirim langsung ke server / coordinator, maka penggunaan energi dalam jaringan akan menjadi boros. Oleh karena itu dalam tugas akhir ini diterapkan metode pemilihan cluster head dalam jaringan untuk mengumpulkan data dari node lain sebelum mengirimkannya ke server / coordinator. Namun, node yang menjadi cluster head mengirimkan data dalam jumlah besar sehingga node cepat kehabisan energi dan mengakibatkan energi dalam jaringan menjadi tidak seimbang. Dikarenakan besarnya penggunaan energi tersebut, maka diterapkan metode round-robin untuk memilih node cluster head yang bergantian. Selain jaringan yang

dibuat, penulis juga membuat sebuah aplikasi antarmuka untuk menampilkan simulasi pergantian cluster head dan menampilkan data yang dikirim.

Pada tugas akhir ini, penulis menguji coba sistem dari sisi fungsionalitasnya dan dari sisi performanya menggunakan scenario yang telah ditentukan. Berdasarkan hasil uji coba, fungsionalitas sistem berjalan dengan baik dan tingkat keberhasilan pengiriman data ketika delay pengiriman 0.5 detik dan 30 data yang diolah sebesar 92.86%, ketika delay pengiriman 1 detik dan 15 data yang diolah, tingkat keberhasilan sebesar 94.01%, dan ketika delay pengiriman 2 detik dan 30 data yang diolah, tingkat keberhasilan sebesar 91.30%.

Kata kunci: Wireless Sensor Network, nRF24l01, Cluster Head, Arduino, Round-robin.

DESIGN AND IMPLEMENTATION OF CLUSTER HEAD SELECTION AND COORDINATION OF SENSOR NODE IN WIRELESS SENSOR NETWORK WITH WIRELESS MODULE NRF24L01

Student's Name	**: MOCH IMAM ZARQONI**
Student's ID	**: 5113100154**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**
Second Advisor	**: Hudan Studiawan, S.Kom., M.Kom.**

Abstract

Wireless Sensor Network (WSN) is a network of nodes that are distributed strategically in a random manner that aims to observe a particular event. Network in this final project is built using wireless module nRF24L01 +. This module is designed for wireless networks that require very low power. It is suitable for use on nodes in wireless sensor networks.

Sensor network in this final project is made in the hope that all nodes can send data to a server / coordinator. However, if all nodes send directly to the server / coordinator, then the use of energy in the network will be wasteful. Therefore in this final project cluster head selection method will be applied in network to collect data from other node before sending it to server / coordinator. However, the cluster head node sends large amounts of data so that the nodes run out of energy faster and cause the energy in the network to become unbalanced. Due to the enormous use of these energies, a round-robin method is used to select alternating cluster head nodes. In addition to the network created, the author also created an interface application to display the simulation of cluster head replacement and display the transmitted data.

In this thesis, the author tested the system from the side of its functionality and in terms of performance using a predetermined scenario. Based on the test results, the system functionality is running well and the success rate of data transmission when the delay of transmission of 0.5 seconds and 30 data processed equal to 92.94%, when delivery delay 1 second and 15 data are processed, the success rate is 94.01%, and when delivery delay 2 seconds and 30 data is processed, the success rate is 91.30%.

Keywords : Wireless Sensor Network, nRF24l01, Cluster Head, Arduino, Round-robin.

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah Swt yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

"**PERANCANGAN DAN IMPLEMENTASI PEMILIHAN *CLUSTER HEAD* DAN KOORDINASI ANTAR SENSOR NODE PADA JARINGAN SENSOR NIRKABEL DENGAN MODUL WIRELESS NRF24L01**"

yang merupakan salah satu syarat dalam menempuh ujian sidang guna memperoleh gelar Sarjana Komputer. Selesainya Tugas Akhir ini tidak terlepas dari bantuan dan dukungan beberapa pihak, sehingga pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Bapak Munthohar Amin (Alm) dan Ibu Siti Fatwiyatun selaku orang tua penulis yang selalu memberikan dukungan doa, moral, dan material yang tak terhingga kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Bapak Waskitho Wibisono, S.Kom., M.Eng., Ph.D. selaku pembimbing I dan dosen wali saya yang telah membimbing dan memberikan motivasi, nasehat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
3. Bapak Hudan Studiawan, S.Kom., M.Kom. selaku II yang telah membimbing dan memberikan motivasi, nasehat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
4. Bapak Darlis Herumurti, S.Kom., M.Kom. selaku kepala jurusan Teknik Informatika ITS.
5. Seluruh dosen dan karyawan Teknik Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa studi di ITS.

6. Ibu Eva Mursidah dan Ibu Sri Budiati yang selalu mempermudah penulis dalam peminjaman buku di RBTC.
7. Teman-teman Keluarga Muslim Informatika, yang sudah banyak meluruskan penulis.
8. Teman-teman seperjuangan RMK NCC/KBJ, yang telah menemani dan menyemangati penulis.
9. Teman-teman angkatan 2013, yang sudah mendukung saya selama perkuliahan.
10. Sahabat penulis yang tidak dapat disebutkan satu per satu yang selalu membantu, menghibur, menjadi tempat bertukar ilmu dan berjuang bersama-sama penulis.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan sehingga dengan kerendahan hati penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depan.

Surabaya, April 2017

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi komunikasi saat ini telah berkembang dengan sangat pesat, salah satunya dibidang *Wireless Sensor Network* (WSN). WSN adalah sebuah jaringan yang terdiri dari node-node yang didistribusikan baik secara strategis maupun secara random yang bertujuan untuk mengamati suatu kejadian tertentu. WSN sudah dianggap sebagai salah satu penemuan yang sangat penting karena harganya murah dan kuat. **Modul Wireless nRF24L01** adalah sebuah modul komunikasi jarak jauh yang dapat digunakan untuk WSN. Modul memanfaatkan pita gelombang RF 2.4GHz ISM (*Industrial, Scientific and Medical*). Modul ini menggunakan antarmuka SPI (*Serial Peripheral Interface*) untuk berkomunikasi. Modul ini didesain untuk jaringan nirkabel yang membutuhkan daya sangat rendah. Sehingga cocok digunakan pada node-node dalam jaringan sensor network [1].

Pada jaringan sensor yang akan dibuat, diharapkan semua node dapat mengirimkan data ke sebuah server / coordinator. Namun, apabila semua node mengirim langsung ke server / coordinator, maka penggunaan energi dalam jaringan akan menjadi boros. Sehingga perlu adanya node yang menjadi *cluster head* dalam jaringan untuk mengumpulkan data dari node lain sebelum mengirimkannya ke server / coordinator [2]. Namun, node yang menjadi *cluster head* harus mengirimkan data dalam jumlah besar sehingga node akan cepat kehabisan energi dan mengakibatkan energi dalam jaringan menjadi tidak seimbang [3]. Dikarenakan besarnya penggunaan energi tersebut, maka diperlukan sebuah metode pemilihan *cluster head* yang efisien untuk menangani permasalahan tersebut.

Metode dalam melakukan pemilihan *cluster head* diawali dengan menentukan node 1 sebagai *cluster head* pertama. Kemudian setiap node dalam jaringan akan mengirimkan paket yang berisi informasi tingkat energi masing-masing node dan data angka random ke node 1. Kemudian ketika waktu sudah habis, node 1 akan memilih node *cluster head* selanjutnya. Metode dalam memilih node *cluster head* selanjutnya menggunakan metode *round-robin* [4].

Hasil yang diharapkan dari pengerjaan tugas akhir ini adalah berupa jaringan sensor nirkabel yang dapat berkoordinasi secara dinamis dalam menentukan *cluster head* yang dapat menyeimbangkan energi dalam jaringan tersebut.

1.2 Rumusan Masalah

Rumusan masalah yang berusaha diselesaikan dalam tugas akhir ini adalah :

1. Bagaimana membuat sebuah jaringan sensor nirkabel menggunakan modul *wireless* nRF24l01?
2. Bagaimana menerapkan metode pemilihan *cluster head* pada jaringan sensor nirkabel?
3. Bagaimana mengimplementasikan metode *round-robin* dalam pemilihan *cluster head*?

1.3 Batasan Permasalahan

Pada tugas akhir ini ada beberapa batasan yang menjadi pertimbangan, berikut ini batasan-batasan yang menjadi pertimbangan :

1. Jaringan yang digunakan adalah jaringan sensor nirkabel menggunakan modul wireless nRF24L01.
2. Jumlah node yang digunakan sebanyak 4 dengan 1 sebagai server / *coordinator* dan 3 sebagai *end device / cluster head* yang dipilih secara bergantian.
3. *Microcontroller* yang digunakan adalah Arduino.

1.4 Tujuan

Tujuan dari pengerjaan Tugas Akhir ini adalah :

1. Membangun sebuah jaringan sensor nirkabel yang dapat berkoordinasi secara dinamis dalam pemilihan *cluster head*.
2. Membangun sebuah aplikasi antarmuka yang dapat menampilkan simulasi pergantian *cluster head* dan memasukkan data dari node server ke dalam database.

1.5 Manfaat

Hasil dari pengerjaan Tugas Akhir ini memiliki manfaat untuk menghasilkan sebuah jaringan sensor nirkabel yang dapat berkoordinasi secara dinamis dan memiliki penggunaan energi yang seimbang dan tahan lama.

1.6 Metodologi

Pembuatan tugas akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal

Tahap awal tugas akhir ini adalah menyusun proposal tugas akhir. Proposal Tugas Akhir ini berisi tentang perencanaan penerapan metode pemilihan *cluster head* sebagai solusi optimal dari permasalahan dalam pemilihan *cluster head* dan penyeimbangan energy dalam jaringan sensor nirkabel.

1.6.2 Studi Literatur

Studi literatur yang dilakukan dalam pengerjaan Tugas Akhir ini adalah mengenai implementasi metode *cluster head selection* pada jaringan sensor nirkabel untuk menentukan node yang akan dijadikan *cluster head* dalam jaringan. Selain itu, juga

dilakukan studi literatur mengenai implementasi pemilihan *cluster head* menggunakan metode *round-robin* [4]. Sehingga, studi literatur ini dapat diterapkan pada perancangan jaringan sensor nirkabel yang dapat melakukan pemilihan *cluster head* secara dinamis.

1.6.3 Implementasi Metode

Implementasi metode yang dipakai dalam tahapan membangun jaringan ini adalah untuk memilih *cluster head* dari node-node yang ada dengan menggunakan metode *round-robin* agar node-node dapat bergantian menjadi *cluster head* secara berurutan [4]. Jika rentang waktu node *cluster head,* sudah mencapai batas, maka akan dilakukan pemilihan *cluster head* yang baru. Kemudian jika semua node sudah pernah menjadi *cluster head,* maka pengurutan akan dimulai dari awal dengan node pertama kembali menjadi *cluster head.*

1.6.4 Pengujian dan Evaluasi

Pengujian dan evaluasi dari hasil Tugas Akhir ini akan diujicobakan dengan membangun jaringan sensor nirkabel menggunakan modul wireless nRF24L01 yang berlokasi di jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya.

1.6.5 Penyusunan Buku

Pada tahap ini disusun buku sebagai dokumentasi dari pelaksanaan tugas akhir yang mencangkup seluruh konsep, teori, implementasi, serta hasil yang telah dikerjakan.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan tugas akhir adalah sebagai berikut:

1. Bab I. Pendahuluan
   Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan tugas akhir.

2. Bab II. Tinjauan Pustaka
   Bab ini meliputi dasar teori dan penunjang yang berkaitan dengan pokok pembahasan dan mendasari pembuatan Tugas Akhir ini.

3. Bab III. Perancangan Perangkat Lunak dan Perangkat Keras
   Bab ini membahas desain dari jaringan yang akan dibuat meliputi arsitektur dan proses perangkat lunak dan keras.

4. Bab IV. Implementasi
   Bab ini membahas implementasi dari desain jaringan yang akan dilakukan pada tahap desain, meliputi potongan *pseudocode* yang terdapat dalam perangkat lunak dan perangkat keras yang digunakan.

5. Bab V. Hasil Uji Coba dan Evaluasi
   Bab ini membahas uji coba dari jaringan yang dibuat dengan melihat keluaran yang dihasilkan, analisa dan evaluasi untuk mengetahui kemampuan jaringan.

6. Bab VI. Kesimpulan dan Saran
   Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.

7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur dalam tugas akhir.
8. Lampiran.

(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan tugas akhir ini. Pembahasan bertujuan untuk memberikan gambaran secara umum terhadap sistem yang dibuat dan berguna sebagai penunjang dalam pengembangan tugas akhir.

2.1 Mikrokontroler Arduino

Arduino merupakan sebuah mikrokontroler *single-board* yang bersifat *open-source*. Arduino dirancang sedemikian rupa sehingga memudahkan para penggunanya dibidang elektronika. *Board* Arduino didesain menggunakan *processor* Atmel AVR dan mendukung masukan dan keluaran pada *board*-nya [5]. Bahasa pemrograman yang digunakan adalah C/C++. Dalam sebuah mikrokontroler Arduino dapat pula ditanamkan berbagai macam *library* maupun metode selama kapasitas memorinya mencukupi.

2.2 Modul Wireless nRF24L01+

Modul Wireless nRF24L01+ adalah modul komunikasi jarak jauh yang memanfaatkan pita gelombang RF (Radio Frequency) 2.4GHz ISM (*Industrial, Scientific and Medical*) yang didesain untuk jaringan nirkabel yang membutuhkan penggunaan daya sangat rendah. Modul ini berupa sebuah chip *transreceiver* tunggal yang memiliki *baseband logic Enhanced Shockburst.* Modul ini cocok digunakan dengan *Microcontroller* Arduino [6].

nRF24L01+ adalah modul *radio transreceiver* yang hemat daya daripada ZigBee dan memiliki harga yang lebih murah. Jangkauan dari nRF24L01+ sebesar ± 1000 meter sedangkan ZigBee ± 50 meter. Namun node pada ZigBee dikonfigurasi menggunakan AT *command*, atau aplikasi windows yang terpisah. Sedangkan node pada nRF24L01+ dikonfigurasi dengan

meng-*compile firmware* langsung pada EEPROM [6]. Bentuk dari nRF24L01+ dapat dilihat di Gambar 2.1.

Gambar 2.1 Perangkat nRF24L01+

2.3 Arduino IDE

Arduino IDE merupakan sebuah perangkat lunak yang digunakan sebagai tempat untuk menulis logika-logika dari suatu skema rangkaian yang terhubung dengan *board* Arduino. Arduino IDE dibangun dengan bahasa pemrograman Java dan bersifat *cross-platform* [5]. Barisan kode dalam Arduino IDE ditulis mengikuti aturan dari C/C++ dan baris kode ini disebut dengan istilah *sketch*. Gambar 2.2 merupakan halaman antarmuka Arduino IDE.

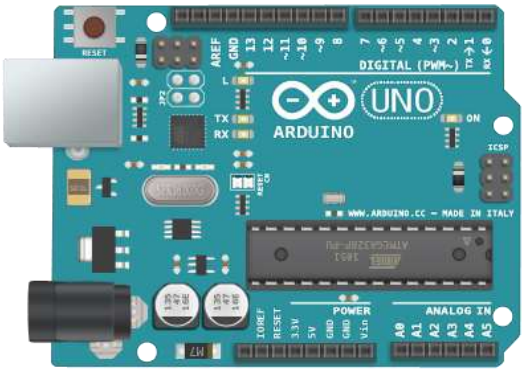
Gambar 2.2 Antarmuka Arduino IDE

2.4 Metode *Round-robin*

Round-robin adalah salah satu algoritma yang digunakan dalam proses komputasi dan *network scheduling*. Dalam algoritma ini, setiap node akan diurutkan dalam sebuah antrian dan setiap node akan mendapat rentang waktu yang sama untuk menjalankan prosesnya. Ketika rentang waktu tersebut habis, maka node selanjutnya akan diberikan rentang waktu yang sama untuk menjalankan prosesnya. Kemudian ketika semua node sudah mendapatkan gilirannya, maka antrian akan diulang dan node pertama akan menjalankan prosesnya lagi. Metode *round-robin* ini mudah diimplementasikan, cukup simple dan meng-*handle* semua node secara setara [4].

2.5 Arduino Uno Revision 3

Arduino Uno adalah salah satu produk berlabel Arduino yang sebenarnya adalah suatu papan elektronik yang mengandung mikrokontroler ATmega328, sebuah keeping yang secara fungsional bertindak seperti sebuah komputer. Piranti ini dapat dimanfaatkan untuk mewujudkan rangkaian elektronik dari yang sederhana hingga yang kompleks [5]. Pengendalian LED hingga pengendalian robot dapat diimplementasikan dengan menggunakan papan yang berukuran cukup kecil ini. Bahkan dengan penambahan komponen tertentu, piranti ini bisa digunakan untuk pemantauan jarak jauh melalui internet. Gambar 2.3 merupakan bentuk dari Arduino Uno R3.



Gambar 2.3 Arduino Uno Revision 3

2.6 MySQL Database

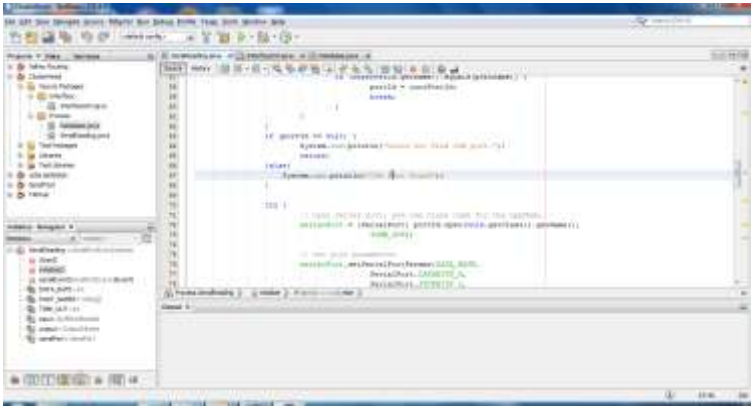
MySQL database adalah sebuah perangkat lunak sistem manajemen basis data SQL atau DBMS yang *multithread, multiuser* dengan sekitar 6 juta instalasi di seluruh dunia. MySQL merupakan pernagkat lunak gratis dibawah licensi GNU General Public License. MySQL dapat berjalan di berbagai sistem operasi. MySQL juga mampu diakses secara bersamaan tanpa adanya konflik dari masing-masing pengguna [7].

2.7 Bahasa Pemrograman *Java*

Bahasa pemrograman *java* adalah sebuah bahasa pemrograman yang berorientasi objek. Bahasa pemrograman ini dirancang agar cukup sederhana sehingga banyak *programmer* dapat menguasai bahasa ini. Bahasa pemrograman *java* berhubungan dengan bahasa pemrograman C dan C++ namun diatur dengan berbeda, dengan sejumlah aspek dari bahasa C dan C++ dihilangkan dan beberapa gagasan dari bahasa lain disertakan [8].

2.8 NetBeans IDE

NetBeans IDE merupakan sebuah perangkat lunak yang digunakan untuk pengembangan aplikasi yang menggunakan bahasa pemrograman *java.* NetBeans IDE juga merupakan aplikasi *cross-platform,* dapat digunakan di sistem operasi Windows, Linux, dan Mac OS [9]. Gambar 2.4 merupakan tampilan antarmuka NetBeans IDE.



Gambar 2.4 Tampilan Antarmuka NetBeans IDE

Netbeans IDE memiliki banyak fitur yang dapat digunakan untuk mempermudah dalam pembangunan sebuah aplikasi

antarmuka. Dalam tugas akhir ini, penulis menggunakan NetBeans IDE untuk membangun aplikasi antarmuka sistem yang menampilkan pergantian *cluster head* dari node-node dalam jaringan.

2.9 Jaringan Sensor Nirkabel

Secara umum, *Wireless Sensor Network* (WSN) didefinisikan sebagai salah satu jenis dari jaringan nirkabel terdistribusi, yang memanfaatkan teknologi *Embeded System* dan seperangkat node sensor, untuk melakukan proses sensor, pengiriman data, monitoring, dan penyajian data ke pengguna melalui komunikasi internet. Setiap jenis sensor memiliki perangkat lunak dan perangkat keras masing-masing yang kemudian akan digabungkan dan dijalankan ke dalam sistem WSN [10].

BAB III
PERANCANGAN PERANGKAT LUNAK

Perancangan merupakan bagian penting dalam pembuatan perangkat lunak dan perangkat keras yang berupa perencanaan secara teknis dari sistem jaringan yang dibuat. Pada bab ini akan dibahas mengenai perancangan dan implementasi metode pemilihan *cluster head* pada sebuah jaringan sensor nirkabel yang dibangun menggunakan Arduino sebagai nodenya dan menggunakan modul *Wireless* nRF4L01+ untuk berkomunikasi antar node.

3.1 Deskripsi Umum

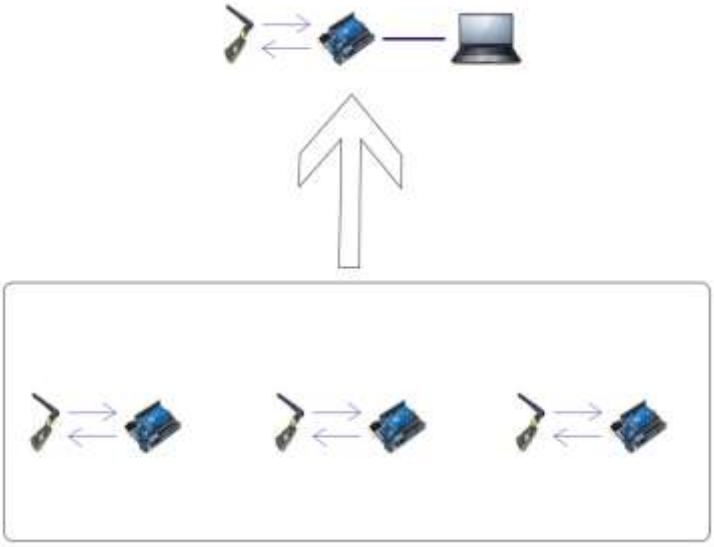
Jaringan sensor nirkabel dalam Tugas Akhir ini akan dibuat menggunakan mikrokontroler Arduino dan modul *Wireless* nRF24L01+. Terdapat 4 node yang akan digunakan untuk membangun jaringan ini. Satu node dijadikan server / coordinator yang bertugas untuk menerima data dari node *cluster head.* 3 node lainnya dijadikan sebagai *end device* / calon *cluster head* dalam jaringan. Node yang menjadi server akan terhubung dengan sebuah komputer, sedangkan node lainnya akan berjalan menggunakan baterai 9 volt sebagai sumber tenaganya. Komunikasi antar node akan dilakukan dengan bantuan *library* RF24Network pada Arduino.

3.2 Arsitektur Umum Sistem

Dalam Tugas Akhir ini akan dilakukan beberapa tahapan untuk membangun jaringan sensor nirkabel. Tahapan awal adalah pembuatan 4 node dengan menggunakan *microcontroller* Arduino dan modul wireless nRF24L01+. Setelah itu, satu node dijadikan sebagai server / *coordinator* yang bertugas menerima data dari node *cluster head.* Kemudian, 3 node lainnya dijadikan sebagai *end device* / calon *cluster head.* Lalu metode *round-robin*

diimplementasikan kepada ketiga node tersebut sehingga node-node tersebut dapat bergantian menjadi *cluster head.* Setelah metode terimplementasikan, kemudian node-node tersebut diletakkan secara strategis dan diaktifkan untuk membentuk sebuah jaringan sensor nirkabel yang dinamis.

Gambar 3.1 menunjukkan ketika node-node pada jaringan baru dipasang, dengan kriteria, satu node menjadi server / *remote coordinator* yang letaknya cukup jauh dari yang lain dan tiga node lainnya menjadi *end device* / calon *cluster head.*



Gambar 3.1 Arsitektur Umum Sistem

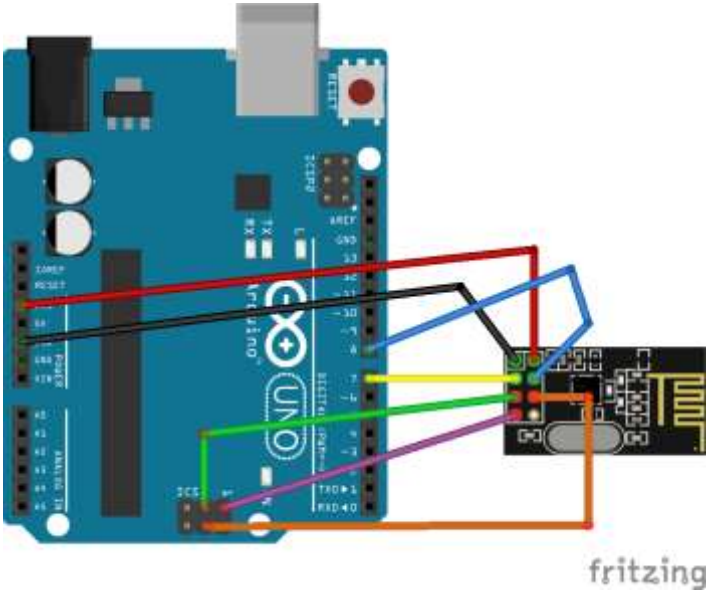
Node yang menjadi *cluster head* akan bertanggung jawab untuk menampung data dari node yang lain dan selanjutnya akan memproses data-data tersebut dan selanjutnya mengirimkannya ke server.

3.3 Perancangan Perangkat Keras

Dalam jaringan yang akan dibuat, node-nodenya merupakan gabungan mikrokontroler Arduino dan modul wireless nRF24L01+.

Pada jaringan ini, perangkat keras yang digunakan untuk membuat node adalah sebagai berikut :

- Satu buah mikrokontroler Arduino Uno R3
- Satu buah modul *wireless* nRF24L01+
- Satu buah baterai 9 V



Gambar 3.2 Rangkaian Perangkat Node

Rangkaian perangkat node ditunjukkan pada Gambar 3.2. pada perangkat node, mikrokontroler dan nRF24L01+ dihubungkan dan kemudian diberi baterai sebagai sumber

dayanya. Untuk *pin* yang menghubungkan Arduino dan nRF24L01+, dijelaskan pada Tabel 3.1.

Tabel 3.1 Koneksi Pin antar Arduino dan nRF24L01+

PIN	Deskripsi
3.3 V	Sumber daya untuk mengaktifkan nRF24L01+
GND	Ground nRF24L01+
D7	SPI Chip Enable
D8	SPI Chip Select
MISO	SPI Slave Data Output, with tri-state option
MOSI	SPI Slave Data Input
SCK	SPI Clock

Pada rangkaian diatas, mikrokontroler Arduino berfungsi sebagai ‘otak’ dari node yang berfungsi untuk mengatur pengiriman dan penerimaan data, pergantian cluster head, dan pengiriman hasil olahan data ke server. Sedangkan nRF24L01+ berfungsi sebagai sarana komunikasi antar node dalam jaringan.

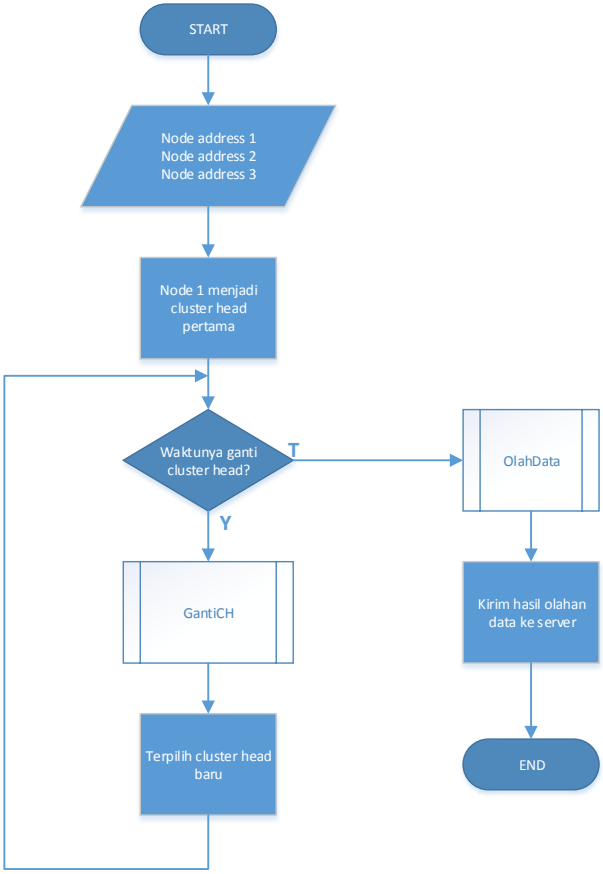
3.4 Diagram Alir Jaringan

Pada subbab ini akan dibahas alur jaringan secara keseluruhan yang melibatkan seluruh node-node yang ada dalam jaringan. Perancangan diagram alir keseluruhan dilakukan untuk lebih dapat memahami dan memberikan pandangan secara menyeluruh mengenai jaringan yang dibangun, serta menunjukkan tentang fungsi-fungsi utama atau proses-proses yang ada dan aliran logika dari jaringan.

Sistem jaringan ini memiliki dua subproses didalamnya. Yaitu GantiCH dan OlahData. Gambar 3.3 menjelaskan garis besar dari alur jalannya sistem jaringan secara keseluruhan beserta subproses didalamnya. Sebelum berjalan, alamat dari masing-masing node harus ditentukan terlebih dahulu. Alamat

dari tiap node ditetapkan berdasarkan urutan dan keberadaannya dalam jaringan.

Ketika sistem mulai berjalan, maka node dengan alamat 1 akan dipilih menjadi *cluster head* pertama. Kemudian, sistem akan melakukan pengecekan berkala apakah waktu node 1 sebagai *cluster head* sudah habis atau belum. Jika waktunya sudah habis, maka subproses GantiCH akan dijalankan. Subproses GantiCH akan menjalankan proses pemilihan *cluster head* baru untuk menggantikan *cluster head* yang sekarang. Sedangkan jika waktu node 1 belum habis, maka subproses OlahData akan dijalankan. Subproses OlahData akan menjalankan proses pengolahan data berupa pencarian nilai minimal, nilai maksimal, dan nilai rata-rata dari data-data yang dikirimkan oleh node-node yang tidak menjadi *cluster head.* Setelah data-data tersebut diolah, maka hasil olahan data tersebut akan dikirimkan ke node server yang terhubung ke sebuah komputer. Gambar 3.3 merupakan diagram alir keseluruhan dari sistem jaringan.

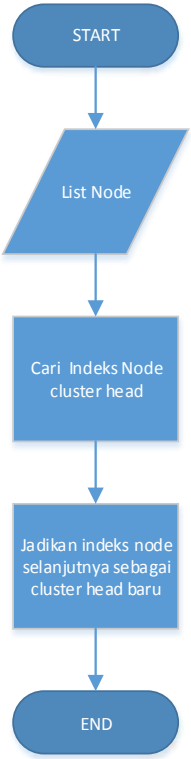


Gambar 3.3 Diagram Alir Sistem

3.4.1 Diagram Alir Subproses GantiCH

Subproses GantiCH merupakan subproses yang bertanggung jawab untuk menentukan *cluster head* baru yang akan menggantikan *cluster head* yang sedang berjalan saat ini. Sebelum subproses ini berjalan, perlu diketahui terlebih dahulu list dari node-node yang ada dalam jaringan. Kemudian, akan

dicari indeks dari node yang sedang menjadi *cluster head* dalam list node tersebut. Maka setelah indeks diketahui, subproses akan memilih node pada indeks setelahnya untuk menjadi *cluster head* baru. Gambar 3.4 merupakan diagram alir dari subproses GantiCH.



Gambar 3.4 Diagram Alir Subproses GantiCH

3.4.2 Diagram Alir Subproses OlahData

Subproses OlahData adalah subproses yang bertanggung jawab untuk mengolah data dari node-node yang ada dalam sistem jaringan. Ketika subproses ini berjalan, data-data dari node

yang diterima oleh node *cluster head* akan diolah untuk mencari data minimal, data maksimal, dan rata-rata data. Gambar 3.5 merupakan diagram alir dari subproses OlahData.

Gambar 3.5 Diagram Alir Subproses OlahData

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi dari perancangan perangkat lunak dan perangkat keras dari sistem jaringan. Cakupan implementasi meliputi proses pengiriman dan penerimaan data, proses pergantian *cluster head*, proses pengolahan data, dan pemasukan data ke database.

4.1 Lingkungan Implementasi

Lingkungan implementasi merupakan suatu lingkungan dimana sistem akan dibangun. Karena sistem merupakan sistem *embedded* pada perangkat mikrokontroler, maka lingkungan implementasi dibagi menjadi dua bagian, yaitu lingkungan implementasi perangkat keras dan implementasi perangkat lunak.

4.1.1 Lingkungan Implementasi Perangkat Keras

Lingkungan implementasi perangkat keras dari sistem yang akan dibangun secara lebih lengkap dijelaskan pada Tabel 4.1 di bawah ini.

Tabel 4.1 Lingkungan Implementasi Perangkat Keras

Perangkat Komputer	**Model :** • ASUS K45VD **Manufaktur :** • ASUSTek Computer Inc. **Processor :** • Intel(R) Core(TM) i3-2370M @2.4 GHz ~2.4 GHz **Memori :** • 4 GB **Sistem Operasi :** • Microsoft Windows 7 Ultimate

	32-bit
Perangkat Mikrokontroler	**Mikrokontroler :** • Atmega328 **Model :** • Arduino UNO *Revision* 3 **Tegangan :** • 5 V **Memori *Flash* :** • 32 KB **SRAM :** • 2 KB ***Transceiver* :** • nRF24L01+

4.1.2 Lingkungan Implementasi Perangkat Lunak

Lingkungan implementasi perangkat lunak dari sistem yang akan dibangun secara lebih lengkap dijelaskan pada Tabel 4.2 di bawah ini.

Tabel 4.2 Lingkungan Implementasi Perangkat Lunak

Perangkat Komputer	**Sistem Operasi :** • Microsoft Windows 7 Ultimate 32-bit ***Software* :** • Arduino IDE 1.6.11 • Netbeans IDE 8.1 **Basis Data :** • MySQL 5.0.11

4.2 Implementasi Perangkat Lunak

Pada bagian implementasi ini akan dibahas mengenai bagaimana implementasi yang dilakukan untuk membangun sistem jaringan. Perangkat lunak yang diimplementasikan terbagi

menjadid tiga bagian, yaitu bagian node klien / *cluster head*, bagian node server, dan bagian tampilan antarmuka. Penjelasan masing-masing bagian akan dijelaskan lebih lengkap pada masing-masing subbab.

4.2.1 Implementasi Bagian Node Klien / *Cluster Head*

Pada bagian node klien / *cluster head*, akan diimplementasikan 5 fungsi termasuk dua subproses yang dijelaskan di bab sebelumnya. Kelima fungsi tersebut adalah *checkIndex* yang berfungsi untuk mencari indeks posisi node dalam list, *olahData* yang berfungsi untuk mengolah data dari node, *sendToSink* yang berfungsi untuk mengirim hasil olahan data ke node server, *sendReceive* yang berfungsi untuk meng-*handle* pergantian mode TX dan mode RX pada node, dan *gantiCH* yang berfungsi untuk melakukan pergantian *cluster head* pada node. Penjelasan masing-masing fungsi akan dijelaskan lebih lengkap pada masing-masing subbab.

4.2.1.1 Inisialisasi Sistem

Proses inisialisasi pada Arduino merupakan suatu hal yang penting. Inisialisasi dilakukan hanya diawal ketika perangkat pertama kali dinyalakan. Inisialisasi yang dilakukan adalah inisialisasi fungsi *setup()* dan variabel global. Inisialisasi fungsi seperti yang ditunjukkan pada Kode Sumber 4.1 dapat dijelaskan melalui beberapa tahapan berikut:

1. Buka port serial dan inisialisasi *baud rate.*
2. Inisialisasi objek “control” sebagai *ThreadController* yang berfungsi untuk meng-*handle multithreading.*
3. Insialisasi *Thread “tSendReceive”* yang bertugas meng-handle mode TX dan RX node.
4. Insialisasi *Thread “tGantiCH”* yang bertugas mengatur pergantian *cluster head.*

5. Insialisasi *Thread "tSendSink"* yang bertugas mengirimkan data ke node server.
6. Inisialisasi pin yang digunakan untuk radio pada RF24 pada objek "radio".
7. Inisialisasi objek "network" untuk jaringan RF24Network dengan dasar objek "radio".
8. Buka port SPI untuk komunikasi nRF24L01+ dengan Arduino.
9. Jalankan nRF24L01+.
10. Jalankan jaringan RF4 dengan *channel* dan alamat node yang sudah ditentukan.
11. Jalankan *Thread* "tSendReceive", "tGantiCH", dan "tSendSink".
12. Masukkan *Thread* "tSendReceive", "tGantiCH", dan "tSendSink" ke dalam objek "control".

```
1  Open Serial.begin(baud rate)
2  Set ThreadController control
3  Set Thread tSendReceive
4  Set Thread tGantiCH
5  Set Thread tSendSink
6  Set RF24 radio(Radio pin)
7  Set RF24Network network(radio)
8  Open SPI.begin
9  Start radio.begin
10 Start network.begin(channel, this_node)
11 Start tSendReceive
12 Start tGantiCH
13 Start tSendSink
14 Add tSendReceive to control
15 Add tGantiCH to control
16 Add tSendSink to control
```

Kode Sumber 4.1 *Pseudocode* inisialisasi awal sistem

Proses inisialisasi variabel global dilakukan di awal sistem berjalan, untuk menentukan variabel-variabel yang dapat digunakan pada keseluruhan sistem. Inisialisasi variabel global

dalam sistem seperti yang ditunjukkan di Kode Sumber 4.2 dapat dijelaskan sebagai berikut :

1. This_node merupakan inisialisasi alamat dari node dimana program dijalankan.
2. First_node merupakan inisialisasi alamat dari node yang menjadi *cluster head.*
3. Count merupakan variabel yang menghitung jumlah node dalam jaringan.
4. Ch merupakan variabel *flag* untuk membedakan antara node *cluster head* dan node biasa.
5. Payload_t merupakan inisialisasi paket pesan dalam jaringan yang dikirim node biasa ke node *cluster head.* Payload_t berisi 3 variabel, yaitu "node" yang berisi node asal paket, "data" yang berisi angka random, dan "persen" yang berisi persentase baterai dari node asal paket.
6. Inisialisasi *array* untuk menampung paket pesan dari semua node dalam jaringan.
7. dataToSink merupakan inisialisasi paket hasil olahan data dari payload_t yang kemudian dikirimkan ke node server. dataToSink berisi 7 variabel, yaitu "chnode" yang berisi alamat node *cluster head,* "persen" yang berisi persentase baterai node, "counter" yang berisi hitungan banyak data yang diolah, "jumlah" yang berisi sum total penjumlahan data, "datarata" yang berisi nilai rata-rata dari data yang diolah, "datamin" yang berisi nilai minimal dari data yang diolah, dan "datamax" yang berisi nilai maksimal dari data yang diolah.

```
1  Set nodeAddress(this_node)
2  Set nodeAddress(first_node)
3  Set count to zero
4  Set ch to one
5  Initialize struct payload_t
6    node
7    Data
8    Persen
9  Set payload_t list_node[11]
```

```
10  Initialize struct dataToSink
11    CHnode
12    Counter
13    Jumlah
14    Datarata
15    Datamin
16    Datamax
```

Kode Sumber 4.2 *Pseudocode* Inisialisasi variabel global

4.2.1.2 Sistem Utama Dalam Fungsi Loop

Proses berjalannya sistem utama terdapat di dalam fungsi *loop()* pada Arduino. Sistem utama dijalankan terus menerus selama node dijalankan. Didalam sistem utama, terdapat 3 perintah yang mengatur jalannya node secara keseluruhan. Yaitu perintah untuk generasi data, perintah untuk *update network*, dan perintah untuk menjalankan kontroler *multithread.* Pada Kode Sumber 4.3 diperlihatkan mengenai jalannya proses *loop* pada sebuah *pseudocode.*

```
1  FUNCTION loop()
2    Set list_node.data = random(1,10)
3    Start network.update()
4    Start control.run()
5  ENDFUNCTION
```

Kode Sumber 4.3 *Pseudocode* Sistem Utama *Loop()*

4.2.1.3 Proses CheckIndex

Proses *checkIndex* merupakan proses pencarian indeks dilakukan ketika sistem ingin mengetahui posisi node dalam *array* list_node.

Pada Kode Sumber 4.4 diperlihatkan mengenai jalannya proses *checkIndex* pada sebuah *pseudocode*. Fungsi ini menggunakan variabel this_node sebagai pembanding untuk mencari indeks node tersebut.

```
1  FUNCTION checkIndex()
2    FOR i = 0 to count
3      IF list_node[i].node = this_node THEN
4        Return i;
5      END IF
6    END FOR
7  ENDFUNCTION
```

Kode Sumber 4.4 *Pseudocode* Fungsi *checkIndex*

4.2.1.4 Proses OlahData

Proses *olahData* merupakan proses yang bertugas untuk mengolah data-data yang dikirim oleh node-node dalam jaringan yang kemudian hasil olahannya akan dikirimkan ke node server.

Pada Kode Sumber 4.5 diperlihatkan mengenai jalannya proses *olahData* pada sebuah *pseudocode*. Fungsi ini mengolah data-data yang ada dalam variabel list_node dan menggunakan *struct* dataToSink untuk menyimpan hasil olahan datanya.

```
 1  Set dataToSink dataAkhir
 2  FUNCTION olahData()
 3    dataAkhir.counter += 1
 4    FOR i = 0 to count
 5      dataAkhir.jumlah += list_node[i].data
 6      IF list_node[i].data > dataAkhir.datamax THEN
 .        dataAkhir.datamax = list_node[i].data
 7      END IF
 8      IF list_node[i].data < dataAkhir.datamin THEN
 9        dataAkhir.datamin = list_node[i].data
10      END IF
11    END FOR
12  ENDFUNCTION
```

Kode Sumber 4.5 *Pseudocode* Fungsi *olahData*

4.2.1.5 Proses sendToSink

Proses *sendToSink* merupakan proses yang bertanggung jawab untuk mengirim data hasil olahan dari fungsi *olahData* ke node server.

Di dalam fungsi *sendToSink*, terdapat tiga tahap yang penting. Yaitu, penentuan alamat node server, pengiriman paket dataAkhir, dan pengkosongan paket dataAkhir setelah paket berhasil dikirim. Proses ini dapat dilihat lebih jelas pada Kode Sumber 4.6.

```
1  FUNCTION sendToSink()
2    Set nodeAddress(sink)
3    Set RF24NetworkHeader header(sink)
4    Start network.write
5          (header,&dataAkhir,sizeof(dataAkhir))
6    dataAkhir = 0
   ENDFUNCTION
```

Kode Sumber 4.6 *Pseudocode* Fungsi *sendToSink*

4.2.1.6 Proses sendReceive

Proses *sendReceive* merupakan proses yang bertanggung jawab untuk meng-*handle* mode TX dan mode RX dari node dengan menggunakan variabel *flag* "ch". Ketika ch bernilai 0, maka node akan menjadi node biasa yang mengirimkan paket ke node *cluster head* secara periodic. Dan ketika ch bernilai 1, maka node akan menjadi *cluster head* yang akan menerima data dari node yang lain. Proses berjalannya fungsi *sendReceive* yang ditunjukkan pada Kode Sumber 4.7 akan dijelaskan secara bertahap sebagai berikut:

1. Mengecek nilai ch untuk mengetahui apakah node sedang menjadi *cluster head* atau tidak.
2. Jika ch bernilai 0, siapkan paket payload untuk dikirim
3. Set objek "header" sebagai RF24NetworkHeader dengan parameter alamat node *cluster head.*
4. Jalankan network.write untuk mengirimkan paket.
5. Jika ch bernilai 1, jalankan pengecekan network.available untuk mengetahui apakah ada pesan yang masuk.
6. Set objek "header" sebagai RF24NetworkHeader.
7. Jalankan network.read untuk membaca paket yang masuk.

8. Lakukan pengulangan sebanyak jumlah node untuk mengecek apakah alamat node sudah terdaftar dalam list_node.
9. Jika sudah, maka *update* data dari alamat node tersebut.
10. Jika belum, maka tambahkan alamat node tersebut ke list_node.

```
1   FUNCTION sendReceive()
2     IF ch = 0 THEN
3       Set payload = {this_node, random(1,10),
4                      vcc.Read_Perc}
5       Set RF24NetworkHeader header(first_node)
6       Start network.write
7             (header,&payload,sizeof(payload))
8     ELSE
9       WHILE network.available
10        Set RF24NetworkHeader header2
11        Set payload_t payload
12        Start network.read
13              (header2,&payload,sizeof(payload))
14        FOR i = 0 to count
15          IF payload.node = list_node[i].node THEN
16            Set List_node[i].data = payload.data
17            Set List_node[i].persen =
18                payload.persen
19            break
20          END IF
21        END FOR
22        IF i = count + 1 THEN
23          Count++
24          Set List_node[count].node = payload.node
25          Set List_node[count].data = payload.data
26          Set List_node[count].persen =
27               payload.persen
28        END IF
29      END WHILE
30      Call olahData()
      END IF
    ENDFUNCTION
```

Kode Sumber 4.7 *Pseudocode* Fungsi *sendReceive*

4.2.1.7 Proses GantiCH

Proses *gantiCH* merupakan proses terpenting dalam sistem ini, dikarenakan proses inilah yang bertugas mengatur pergantian *cluster head.* Fungsi ini memiliki dua proses yang berjalan bergantian sesuai dengan nilai variabel *flag* ch. Ketika ch bernilai 1, maka fungsi ini akan menjalankan proses pergantian *cluster head* pada interval waktu tertentu. Dan ketika ch bernilai 0, maka fungsi akan menjalankan fungsi untuk mencari alamat node yang sedang menjadi *cluster head* pada saat itu.

Fungsi *gantiCH* seperti yang ditunjukkan pada Kode Sumber 4.8 akan dijelaskan lebih lengkap sebagai berikut :

1. Inisialisasi objek “chBaru” untuk mencari *cluster head* baru.
2. Periksa apakah ch bernilai 1 atau 0.
3. Jika ch bernilai 1, maka jalankan fungsi *sort* dengan parameter list_node untuk mengurutkan node dalam jaringan.
4. Set objek “index” untuk mengetahui posisi node *cluster head* saat ini.
5. Set objek “now” untuk pengecekan waktu.
6. Periksa apakah sudah waktunya melakukan pergantian *cluster head.*
7. Jika posisi “index” bukan berada diakhir urutan, maka *cluster head* baru yang dipilih adalah node yang berada diurutan selanjutnya. Jika posisi “index” berada diakhir urutan, maka *cluster head* baru adalah node diurutan pertama.
8. Kirimkan pemberitahuan alamat node *cluster head* baru ke semua node dalam jaringan.
9. Ubah ch menjadi 0.
10. Jika ch bernilai 0, maka lakukan pengecekan berkala untuk mengetahui alamat node *cluster head.*

```
1  FUNCTION gantiCH()
```

```
2     Set integer chBaru;
3     IF ch = 1 THEN
4       Call sort(list_node)
5       set index to checkIndex()
6       set now to millis()
7       IF now-waktu >= 60000 * persen / 100 AND
8       count != 0 THEN
9         IF index < count THEN
10          Set chBaru = index + 1
11        ELSE
12          Set chBaru = 0
13        ENDIF
14        Set First_node = list_node[chBaru].node
15        FOR i = 0 to count
16          IF i != index THEN
17            Set RF2NetworkHeader
18                  header3(list_node[i].node)
19            Start network.write
20                  (header3,&first_node,sizeof(fi
21                  rst_node))
22          END IF
23        END FOR
24        Set ch = 0
25      END IF
26    ELSE
27      WHILE network.available
28        Set RF2NetworkHeader header4
29        Set temp
30        Start network.read
31              (header4,&temp,sizeof(temp))
32        Set first_node = temp
33        IF this_node = first_node THEN
34          Set ch = 1
35          Set waktu = millis()
36        END IF
37      END WHILE
    END IF
ENDFUNCTION
```

Kode Sumber 4.8 *Pseudocode* Fungsi *gantiCH*

4.2.2 Implementasi Bagian Node Server / Sink

Pada bagian ini, akan dijelaskan mengenai implementasi perangkat lunak yang dilakukan untuk node server / *sink*. Node server ini berfungsi untuk menerima data hasil olahan yang dikirimkan oleh node yang menjadi *cluster head.* Terdapat dua implementasi penting yang harus dilakukan agar node server ini dapat berjalan. Penjelasan selanjutnya akan dibahas di masing-masing subbab.

4.2.2.1 Inisialisasi Sistem

Proses inisialisasi pada Arduino merupakan suatu hal yang penting. Inisialisasi dilakukan hanya diawal ketika perangkat pertama kali dinyalakan. Inisialisasi yang dilakukan adalah inisialisasi fungsi *setup()* dan variabel global. Inisialisasi fungsi seperti yang ditunjukkan pada Kode Sumber 4.9 dapat dijelaskan melalui beberapa tahapan berikut:

1. Buka port serial dan inisialisasi *baud rate.*
2. Inisialisasi objek "radio" sebagai radio pada RF24 dengan parameter pin radio pada Arduino.
3. Inisialisasi objek "network" sebagai jaringan RF24Network dengan dasar objek "radio".
4. Buka port SPI untuk komunikasi antara nRF24L01+ dengan Arduino.
5. Jalankan radio nRF24L01+.
6. Jalankan jaringan RF24Network dengan *channel* dan alamat node yang sudah ditentukan.

```
1 Open Serial.begin(baud rate)
2 Set RF24 radio(Radio pin)
3 Set RF24Network network(radio)
4 Open SPI.begin
5 Start radio.begin
6 Start network.begin(channel, this_node)
```

Kode Sumber 4.9 *Pseudocode* Inisialisasi awal sistem

Proses inisialisasi variabel global dilakukan di awal sistem berjalan, untuk menentukan variabel-variabel yang dapat digunakan pada keseluruhan sistem. Inisialisasi variabel global dalam sistem seperti yang ditunjukkan di Kode Sumber 4.10 dapat dijelaskan sebagai berikut :

1. Inisialisasi alamat node server dengan variabel this_node.
2. dataToSink merupakan inisialisasi paket hasil olahan data dari payload_t yang kemudian dikirimkan ke node server. dataToSink berisi 7 variabel, yaitu "chnode" yang berisi alamat node *cluster head*, "persen" yang berisi persentase baterai node, "counter" yang berisi hitungan banyak data yang diolah, "jumlah" yang berisi sum total penjumlahan data, "datarata" yang berisi nilai rata-rata dari data yang diolah, "datamin" yang berisi nilai minimal dari data yang diolah, dan "datamax" yang berisi nilai maksimal dari data yang diolah.

```
1  Set nodeAddress(this_node)
2  Initialize struct dataToSink
3    CHnode
4    Persen
5    Counter
6    Jumlah
7    Datarata
8    Datamin
9    Datamax
```

Kode Sumber 4.10 *Pseudocode* Inisialisasi variabel global

4.2.2.2 Sistem Utama dalam Fungsi Loop

Proses berjalannya sistem utama terdapat di dalam fungsi *loop()* pada Arduino. Sistem utama dijalankan terus menerus selama node dijalankan. Didalam sistem utama, terdapat 4 perintah yang mengatur jalannya node secara keseluruhan. Yaitu perintah untuk *update network*, perintah untuk mengecek keberadaan jaringan, perintah untuk membaca paket masuk, dan

perintah untuk menampilkan data hasil olahan. Pada Kode Sumber 4.11 diperlihatkan mengenai jalannya proses *loop* pada sebuah *pseudocode.*

```
1  FUNCTION loop()
2    Start network.update()
3    WHILE network.available
4      Set RF2NetworkHeader header
5      Set dataToSink payload_t
6      Start network.read
7             (header,&payload_t,&sizeof(payload_t))
8      Print payload_t
9    END WHILE
10 ENDFUNCTION
```

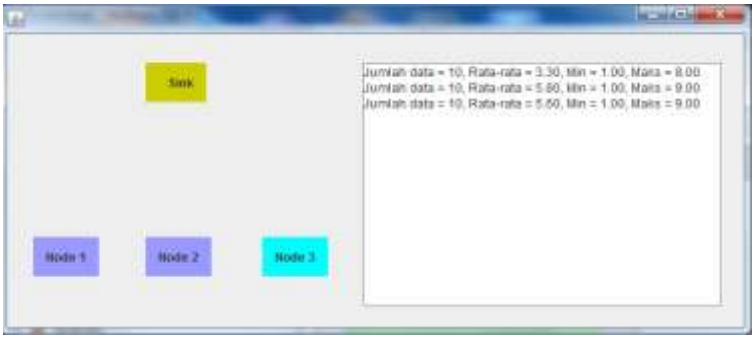
Kode Sumber 4.11 *Pseudocode* Sistem Utama Fungsi *Loop()*

4.2.3 Implementasi Antarmuka Sistem

Pada bagian ini, akan dijelaskan mengenai implementasi perangkat lunak yang dilakukan untuk membuat antarmuka dari sistem. Tampilan antarmuka ini berfungsi untuk mensimulasikan pergantian *cluster head*, menampilkan data yang diterima node server, dan memasukkan data ke dalam database. Terdapat 3 bagian implementasi penting yang harus dilakukan agar antarmuka dapat berjalan. Penjelasan selanjutnya akan dibahas di masing-masing subbab.

4.2.3.1 Tampilan Antarmuka

Tampilan antarmuka merupakan sebuah halaman GUI (*Graphical User Interface*) yang dibangun menggunakan bahasa pemrograman Java dan IDE Netbeans. Halaman ini menampilkan simulasi pergantian *cluster head* dan menampilkan hasil olahan data yang diterima node server. Gambar 4.1 merupakan implementasi tampilan antarmuka sistem.



Gambar 4.1 Implementasi Tampilan Antarmuka Sistem

4.2.3.2 Class SerialReading

Class *SerialReading* merupakan salah satu *class* dalam pembuatan tampilan antarmuka sistem. *Class* ini berfungsi untuk membaca data dari node server melalui port serial untuk selanjutnya ditampilkan di tampilan antarmuka sistem. Pada Kode Sumber 4.12 diperlihatkan secara jelas isi dari *class SerialReading* pada sebuah *pseudocode.* Mengenai bagian-bagian dari *class SerialReading,* akan dijelaskan sebagai berikut :

1. Deklarasikan nama *class.*
2. Inisialisasi port serial yang akan dibaca.
3. Inisialisasi objek untuk menampung input dan output.
4. Fungsi *initialize()* untuk memulai membaca port serial.
5. Fungsi *close()* untuk menutup port setelah selesai dibaca.
6. Fungsi *serialEvent()* untuk menampung data dari port serial ke variabel yang nantinya akan ditampilkan di tampilan antarmuka dan juga dikirim ke database.

```
1  class SerialReading implements
2                        SerialPortEventListener
3    SerialPort serialPort
4    String PORT_NAMES[] := { "/dev/tty.usbserial-
5                             A9007UX1",
6                             "/dev/ttyACM0",
7                             "/dev/ttyUSB0", "COM22"
```

```
 8                                }
 9
10    BufferedReader input
11    OutputStream output
12    TIME_OUT := 2000
13    DATA_RATE := 57600
14
15    Call function initialize()
16    Call function close()
17    Call function serialEvent()
.
18  END Class
```

Kode Sumber 4.12 *Pseudocode class SerialReading*

Seperti yang dijelaskan sebelumnya, didalam *class SerialReading* ini terdapat 3 fungsi penting yang harus diperhatikan. Fungsi yang pertama adalah fungsi *initialize.* Fungsi ini berfungsi untuk membaca port serial dari arduino. Kode Sumber 4.13 menunjukkan *pseudocode* dari fungsi *initialize.*

```
 1  function initialize()
 2      CommPortIdentifier portId := NIL
 3      Enumeration portEnum :=
 4                   CommPortIdentifier.getPortIdent
 5                   ifiers()
 6      while (portEnum.hasMoreElements())
 7        CommPortIdentifier currPortId :=
 8                              (CommPortIdentifier)
 9                              portEnum.nextElement(
10                              )
11      for each String portName in PORT_NAMES
.
12  if(currPortId.getName().equals(portName))
13      portId := currPortId
14            break
15      if(portId == NIL)
16        writeln "Could not find COM port."
17        return
18      try
19        serialPort := (SerialPort)
20                       portId.open(this.getClass()
21                       .getName(), TIME_OUT)
```

```
22        serialPort.setSerialPortParams(DATA_RATE,
23                                SerialPort.DATABITS_8,
24                                SerialPort.STOPBITS_1,
25                                SerialPort.PARITY_NONE)
26        input := new BufferedReader(new
27                 InputStreamReader(serialPort.getIn
28                 putStream()))
29        output := serialPort.getOutputStream()
30        serialPort.addEventListener(this)
31        serialPort.notifyOnDataAvailable(true)
32      catch (Exception e)
33        System.err.println(e.toString())
34  ENDFUNCTION
```

Kode Sumber 4.13 *Pseudocode* fungsi *initialize* dalam *class SerialReading*

Fungsi penting yang kedua dalam *class SerialReading* adalah fungsi *close* yang berfungsi untuk menutup port setelah selesai dibaca. Kode Sumber 4.14 menunjukkan *pseuducode* dari fungsi *close.*

```
1  synced function close()
2      if(serialPort != NIL)
3        serialPort.removeEventListener()
4        serialPort.close()
5  ENDFUNCTION
```

Kode Sumber 4.14 *Pseudocode* fungsi *close* dalam *class SerialReading*

Fungsi yang ketiga dan yang paling penting dalam *class SerialReading* adalah fungsi *serialEvent.* Fungsi ini bertugas untuk mengolah data yang dibaca dari port serial untuk kemudian ditampilkan di antarmuka dan dimasukkan ke dalam database. Kode Sumber 4.15 menunjukkan *pseudocode* dari fungsi *serialEvent.*

```
1  synced function serialEvent(SerialPortEvent
2                                          oEvent)
3      if(oEvent.getEventType() ==
```

```
4                     SerialPortEvent.DATA_AVAILABLE)
5       try
          String inputLine := input.readLine()
          String[] data := inputLine.split(",")
          InterfaceCH.gantiCH(Integer.parseInt(data
          [0]))
          String tampil := "Jumlah data = " +
                  data[1] + ", Rata-rata = " +
                  data[2] + ", Min = " + data[3] +
                  ", Maks = " + data[4]
          InterfaceCH.cariline(tampil)
          Database.InsertToDB(data[0],
                          Integer.parseInt(data[1])
                          ,
                          Float.parseFloat(data[2])
                          ,
                          Float.parseFloat(data[3])
                          ,
                          Float.parseFloat(data[4])
                          )
          writeln inputLine
        catch (Exception e)
          System.err.println(e.toString())
ENDFUNCTION
```

Kode Sumber 4.15 *Pseudocode* fungsi *serialEvent* dalam *class SerialReading*

4.2.3.3 Class Database

Class ini digunakan untuk melakukan koneksi ke database MySQL dan juga untuk memasukkan data dari node server ke dalam database. Pada Kode Sumber 4.16 diperlihatkan secara jelas isi dari *class Database* pada sebuah *pseudocode.* Mengenai bagian-bagian dari *class Database,* akan dijelaskan sebagai berikut :

1. Deklarasikan *class Database*.
2. Inisialisasi objek untuk koneksi ke database.
3. Fungsi *ConInit()* untuk melakukan koneksi ke database dengan user dan password yang sudah dibuat.

4. Fungsi *InsertToDB()* untuk memasukkan data ke dalam table yang sudah ditentukan.

```
1  class Database
2      Statement stmt
3      Connection con
4      ResultSet rs
5
6      Call function ConInit()
7      Call function InsertToDB()
```

Kode Sumber 4.16 *Pseudocode* untuk *Class Database*

Seperti yang dijelaskan sebelumnya, didalam *class Database* ini terdapat dua fungsi yang perlu diperhatikan. Yang pertama adalah fungsi *ConInit.* Fungsi ini bertugas untuk membuat koneksi antara aplikasi dengan basis data MySQL yang telah dibuat. Kode Sumber 4.17 menunjukkan *pseudocode* dari fungsi *ConInit.*

```
1  function ConInit() throws ClassNotFoundException,
2                           SQLException
3          Class.forName("com.mysql.jdbc.Driver")
4          con := DriverManager.getConnection
5                  ("jdbc:mysql://localhost/ta_clusterh
6                  ead?zeroDateTimeBehavior=convertToNu
7                  ll", "root", "")
8          stmt := con.createStatement()
9  ENDFUNCTION
```

Kode Sumber 4.17 *Pseudocode* fungsi ConInit dalam *class Database*

Kemudian fungsi berikutnya dalam *class Database* adalah fungsi *InsertToDB.* Fungsi ini bertugas untuk memasukkan data ke dalam table dalam basis data MySQL yang telah dibuat. Kode Sumber 4.18 menunjukkan *pseudocode* dari fungsi *InsertToDB.*

```
1  function InsertToDB(String node,j,r,dmin,dmax)
2                              throws SQLException
```

```
3       stmt.executeUpdate("INSERT INTO datanode
4                          (ch_node,jumlah,
5                          rata_rata,
6                          datamin,datamax) VALUES "
7                          + "('" + node + "'," + j
8                          + "," + r + "," + dmin +
9                          "," + dmax + ")")
10  ENDFUNCTION
```

Kode Sumber 4.18 *Pseudocode* fungsi *InsertToDB* dalam *class Database*

BAB V
HASIL UJI COBA DAN EVALUASI

Pada bab ini akan dijabarkan mengenai uji coba dan evaluasi Tugas Akhir yang telah dikerjakan. Sistem akan diuji coba fungsionalitas dan performanya dengan menjalankan serangkaian skenario pengujian yang telah ditentukan. Hasil evaluasi menjelaskan tentang rangkuman dari hasil pengujian, yang akan dijelaskan pada akhir bab ini.

5.1 Lingkungan Uji Coba

Lingkungan uji coba mencakup perangkat dan peralatan apa saja yang digunakan dalam uji coba. Uji coba fungsionalitas dilakukan di laboratorium Komputasi Berbasis Jaringan di Kampus Teknik Informatika ITS dan uji coba performa dilakukan di lapangan yang terletak di kompleks Blok U ITS. Lingkungan uji coba memiliki spesifikasi seperti di bawah ini :

- Laptop ASUS K45VD
  - Intel(R) Core(TM) i3-2370M @2.40 GHz ~2.4 GHz
  - RAM 4 GB
  - Microsoft Windows 7 Ultimate 32-bit
- Empat node dengan setiap node memiliki komponen sebagai berikut :
  - Mikrokontroler Arduino UNO R3
  - Modul *transceiver* nRF24L01+
  - Kabel USB
  - Baterai 9 V
  - Satu set kabel jumper

5.2 Uji Coba Fungsionalitas

Uji coba fungsionalitas merupakan sebuah pengujian yang dilakukan untuk menguji bekerja atau tidaknya fungsi – fungsi utama yang ada pada sistem. Fungsionalitas yang diuji dalam bagian ini yaitu fungsionalitas dari setiap bagian sistem. Bagian

yang dimaksud meliputi node klien / *cluster head*, node server, dan tampilan antarmuka sistem.

5.2.1 Uji Coba Fungsionalitas Node Klien / *Cluster Head*

Bagian pertama dari uji coba fungsionalitas adalah uji coba fungsionalitas pada node klien / *cluster head*. Fungsionalitas yang akan diuji mencakup pengolahan data, pengiriman data ke node server, dan pergantian *cluster head*.

5.2.1.1 Uji Coba Pengolahan Data

Uji coba ini dijalankan dengan tujuan untuk menguji fungsionalitas pengolahan data dari node *cluster head*. Pada uji coba ini, node *cluster head* akan mengumpulkan 5 gelombang data yang berupa angka random dari 3 node di jaringan lalu kemudian mengolahnya untuk mendapatkan data rata-rata, data minimal, dan data maksimal. Tabel 5.1 adalah tabel mengenai skenario uji coba pengolahan data pada node *cluster head*.

Tabel 5.1 Skenario Uji Coba Pengolahan Data Pada Node *Cluster Head*

ID	**UJ-F01**
Nama	Uji Coba Pengolahan Data Pada Node *Cluster Head*
Tujuan Uji Coba	Menguji fungsionalitas node untuk mengolah data dari node-node lain di jaringan.
Kondisi Awal	Semua node terhubung ke jaringan
Skenario	1. Menentukan node *cluster head* 2. Mengumpulkan data dari node-node klien 3. Menampilkan hasil olahan data
Masukan	Data dari node-node klien
Keluaran	Data rata-rata, data minimal, data maksimal
Hasil yang Diharapkan	Tampilan yang menampilkan data rata-rata, data minimal, dan data maksimal yang sesuai dengan data masukan

5.2.1.2 Uji Coba Pengiriman Hasil Olah Data ke Node Server

Uji coba ini dijalankan dengan tujuan untuk menguji fungsionalitas pengiriman hasil olah data ke node server. Pada uji coba ini, node *cluster head* setelah melakukan pengolahan data akan mengirimkan hasil olah data tersebut ke node server. Tabel 5.2 adalah tabel mengenai skenario uji coba pengiriman hasil olah data ke node server.

Tabel 5.2 Skenario Uji Coba Pengiriman Hasil Olah Data ke Node Server

ID	**UJ-F02**
Nama	Uji Coba Pengiriman Hasil Olah Data ke Node Server
Tujuan Uji Coba	Menguji fungsionalitas node *cluster head* dalam mengirimkan data ke node server.
Kondisi Awal	Semua node terhubung ke jaringan
Skenario	1. Node *cluster head* mengolah data dari node klien 2. Node *cluster head* mengirimkan data ke node server 3. Node server menampilkan paket data yang diterima
Masukan	Data hasil olahan
Keluaran	Node server menampilkan data hasil olahan
Hasil yang Diharapkan	Data berhasil dikirim ke node server

5.2.1.3 Uji Coba Pergantian *Cluster Head*

Uji coba ini dijalankan dengan tujuan untuk menguji fungsionalitas pergantian node *cluster head* antar node dalam jaringan. Pada uji coba ini, node *cluster head* akan memilih penggantinya dengan menggunakan metode *round-robin*. Tabel 5.3 adalah tabel mengenai skenario uji coba mendaftarkan sensor baru.

Tabel 5.3 Skenario Uji Coba Pergantian *Cluster Head*

ID	**UJ-F03**
Nama	Uji Coba Pergantian *Cluster Head*
Tujuan Uji Coba	Menguji fungsionalitas node untuk melakukan pergantian *cluster head*.
Kondisi Awal	Semua node terhubung ke jaringan
Skenario	1. Node 1 menjadi *cluster head* 2. Setelah berjalan selama waktu yang ditentukan, maka akan melakukan pergantian *cluster head* 3. Node 2 terpilih menjadi *cluster head* 4. Setelah berjalan selama waktu yang ditentukan, maka akan melakukan pergantian *cluster head* 5. Node 3 terpilih menjadi *cluster head*
Masukan	List node dalam jaringan
Keluaran	-
Hasil yang Diharapkan	Pergantian *cluster head* berjalan dengan lancer dan semua node mendapat giliran

5.2.2 Uji Coba Fungsionalitas Node Server

Pembahasan berikutnya adalah uji coba pada sisi node server. Fungsionalitas yang diuji pada bagian ini adalah fungsi menerima data hasil olahan yang dikirim node *cluster head*.

Uji coba ini dijalankan dengan tujuan untuk menguji fungsionalitas penerimaan paket data pada node server. Pada uji coba ini, node server akan menerima dan menampilkan paket data yang dikirimkan oleh node *cluster head*. Tabel 5.4 adalah tabel mengenai skenario uji coba.

Tabel 5.4 Skenario Uji Coba Menerima Paket Data dari Node *Cluster Head*

ID	**UJ-F04**
Nama	Uji Coba Menerima Paket Data dari Node *Cluster Head*
Tujuan Uji Coba	Menguji fungsionalitas node server dalam menerima paket data dari node *cluster head*.
Kondisi Awal	Semua node terhubung ke jaringan

Skenario	1. Node *cluster head* mengirimkan data ke node server 2. Dilakukan pemantauan apakah pengiriman berhasil dilakukan 3. Jika iya, kemudian dilakukan pemantauan apakah paket diterima dengan benar oleh node server 4. Melihat apakah node server menampilkan data atau tidak
Masukan	Paket dari node *cluster head*
Keluaran	Tampilan node server yang menunjukkan data yang sudah berhasil diterima
Hasil yang Diharapkan	Node server dapat menerima paket data dengan benar

5.2.3 Uji Coba Fungsionalitas Tampilan Antarmuka Sistem

Pembahasan berikutnya adalah uji coba pada sisi tampilan antarmuka sistem. Fungsionalitas yang diuji pada bagian ini adalah fungsi membaca port serial, fungsi menampilkan data, dan fungsi memasukkan data ke dalam database.

5.2.3.1 Uji Coba Membaca Port Serial

Uji coba ini dijalankan dengan tujuan untuk menguji fungsionalitas membaca port serial yang dilakukan oleh tampilan antarmuka sistem. Pada uji coba ini, aplikasi antarmuka sistem akan membaca data dari port serial milik node server untuk kemudian menampilkannya. Tabel 5.5 adalah tabel mengenai skenario uji coba.

Tabel 5.5 Skenario Uji Coba Membaca Port Serial

ID	**UJ-F05**
Nama	Uji Coba Membaca Port Serial
Tujuan Uji Coba	Menguji fungsionalitas aplikasi antarmuka sistem dalam membaca port serial milik node server.
Kondisi Awal	Membuka aplikasi antarmuka
Skenario	1. Menghubungkan node server ke aplikasi

	antarmuka 2. Menentukan nomor port serial yang dibaca 3. Memantau apakah port serial dapat dibaca atau tidak 4. Jika iya, baca port serial dan tampilkan di layar
Masukan	Data dari port serial node server
Keluaran	Tampilan data yang diterima oleh aplikasi antarmuka
Hasil yang Diharapkan	Aplikasi antarmuka dapat membaca data dari port serial milik node server

5.2.3.2 Uji Coba Menampilkan Data

Uji coba ini dijalankan dengan tujuan untuk menguji fungsionalitas menampilkan data yang dilakukan oleh tampilan antarmuka sistem. Pada uji coba ini, aplikasi antarmuka sistem akan menampilkan data dari port serial yang sudah berhasil dibaca ke dalam layar. Tabel 5.6 adalah tabel mengenai skenario uji coba.

Tabel 5.6 Uji Coba Menampilkan Data

ID	**UJ-F06**
Nama	Uji Coba Menampilkan Data
Tujuan Uji Coba	Menguji fungsionalitas aplikasi antarmuka sistem dalam menampilkan data yang diterima dari port serial.
Kondisi Awal	Membuka aplikasi antarmuka
Skenario	5. Membaca port serial 6. Memantau apakah aplikasi menampilkan data dengan benar
Masukan	Data dari port serial node server
Keluaran	Tampilan data yang diterima oleh aplikasi antarmuka
Hasil yang Diharapkan	Aplikasi antarmuka dapat menampilkan data di layar

5.2.3.3 Uji Coba Memasukkan Data ke Database

Uji coba ini dijalankan dengan tujuan untuk menguji fungsionalitas memasukkan data yang diterima oleh aplikasi antarmuka ke dalam database MySQL. Pada uji coba ini, aplikasi antarmuka sistem akan memasukkan data yang diterima ke dalam database MySQL untuk disimpan. Tabel 5.7 adalah tabel mengenai skenario uji coba.

Tabel 5.7 Uji Coba Memasukkan Data ke Database

ID	**UJ-F07**
Nama	Uji Coba Memasukkan Data ke Database
Tujuan Uji Coba	Menguji fungsionalitas aplikasi antarmuka sistem dalam memasukkan data ke database MySQL.
Kondisi Awal	Membuka aplikasi antarmuka
Skenario	7. Membaca port serial 8. Memantau apakah aplikasi menampilkan pesan error dalam memasukkan data ke database 9. Jika tidak, lakukan pemantauan pada database yang telah dibuat apakah data berhasil masuk atau tidak
Masukan	Data dari port serial node server
Keluaran	-
Hasil yang Diharapkan	Aplikasi antarmuka dapat memasukkan data ke dalam database dengan benar

5.3 Hasil dan Evaluasi Uji Coba Fungsionalitas

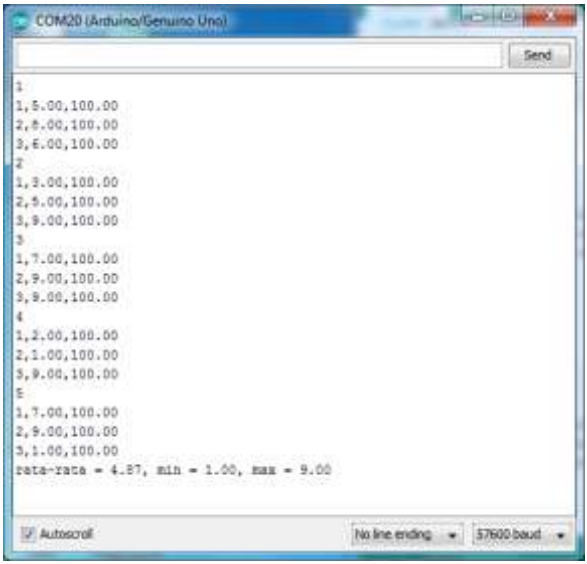
Pada bagian ini, akan dijelaskan mengenai hasil dari masing-masing uji coba fungsionalitas yang telah dilakukan. Skenario uji coba yang telah dijabarkan terdiri dari beberapa hal. Secara garis besar, pengujian terhadap fungsionalitas terbagi menjadi tiga yaitu fungsionalitas node klien / *cluster head*, node server, dan aplikasi antarmuka sistem.

5.3.1 Hasil Uji Coba Fungsionalitas Node Klien / *Cluster Head*

Berdasarkan uji coba yang telah dilakukan untuk menguji fungsionalitas node klien / *cluster head*, berikut ini hasil dari uji coba yang dilakukan :

5.3.1.1 Hasil Uji Coba Pengolahan Data

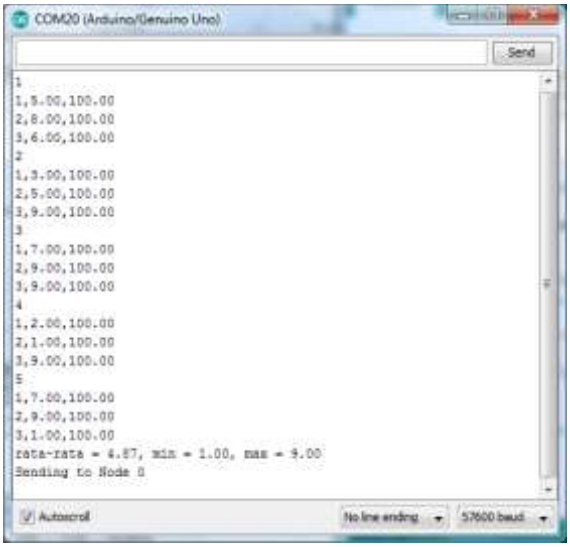
Menurut skenario pada Tabel 5.1, yang pertama kali dilakukan adalah menentukan node *cluster head*. Setelah node *cluster head* terpilih, maka node akan mengumpulkan data dari node-node klien dalam jaringan. Kemudian, setelah terkumpul lima gelombang data, maka node *cluster head* akan mengolah data untuk mendapatkan rata-rata, minimal dan minimal dari data. Pada uji coba yang telah dilakukan, didapatkan hasil seperti pada Gambar 5.1. Hal ini menunjukkan bahwa fungsionalitas pengolahan data bekerja dengan baik.



Gambar 5.1 Hasil Uji Coba Pengolahan Data

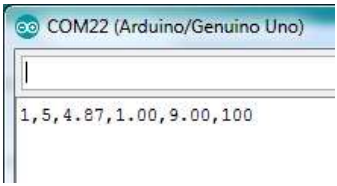
5.3.1.2 Hasil Uji Coba Pengiriman Hasil Olah Data ke Node Server

Menurut skenario pada Tabel 5.2, yang pertama kali dilakukan adalah node *cluster head* menjalankan fungsi pengolahan data. Setelah data berhasil diolah, maka data dimasukkan ke dalam sebuah paket. Kemudian, paket tersebut dikirimkan ke node server. Pada uji coba yang telah dilakukan, didapatkan hasil seperti pada Gambar 5.2. Hal ini menunjukkan bahwa fungsionalitas pengiriman hasil olah data ke node server bekerja dengan baik.



Gambar 5.2 Hasil Uji Coba Pengiriman Hasil Olah Data ke Node Server

Setelah paket dikirimkan oleh node *cluster head*, selanjutnya kita harus memeriksa dari sisi node server apakah data diterima atau tidak. Gambar 5.3 menunjukkan tampilan dari sisi server bahwa data sudah berhasil diterima.



Gambar 5.3 Hasil Uji Coba Pengiriman Data ke Node Server di Sisi Node Server

5.3.1.3 Hasil Uji Coba Pergantian *Cluster Head*

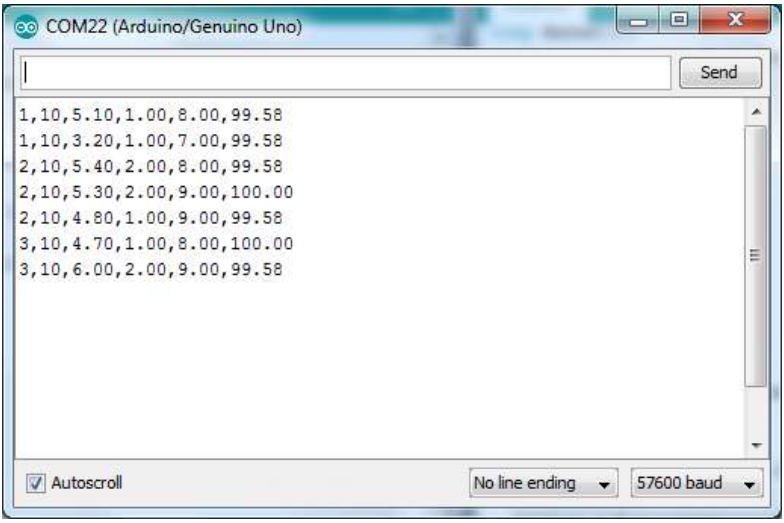
Menurut skenario pada Tabel 5.3, yang pertama kali dilakukan adalah menentukan node 1 sebagai node *cluster head*. Setelah itu dilakukan pengecekan berkala untuk mengecek apakah sudah waktunya ganti. Jika waktu pergantian sudah dating, maka node 1 akan menjalankan fungsi pergantian untuk memilih *cluster head* baru. Pada uji coba yang telah dilakukan, didapatkan hasil seperti pada Gambar 5.4. Hal ini menunjukkan bahwa fungsionalitas pergantian node *cluster head* berjalan dengan baik.

Gambar 5.4 Hasil Uji Coba Pergantian Node *Cluster Head*

5.3.2 Hasil Uji Coba Fungsionalitas Node Server

Berdasarkan uji coba yang telah dilakukan untuk menguji fungsionalitas node server, berikut ini hasil dari uji coba yang dilakukan.

Menurut skenario pada Tabel 5.4, yang pertama kali dilakukan adalah node 1 sebagai node *cluster head* mengirimkan data hasil olahan ke node server. Setelah itu dilakukan pemantauan apakan data berhasil dikirim atau tidak. Kemudian, data yang berhasil diterima akan ditampilkan oleh node server. Pada uji coba yang telah dilakukan, didapatkan hasil seperti pada Gambar 5.5. Hal ini menunjukkan bahwa fungsionalitas penerimaan paket data dari node *cluster head* berjalan dengan baik.



Gambar 5.5 Node Server Menerima Paket dari Node *Cluster Head*

5.3.3 Hasil Uji Coba Fungsionalitas Aplikasi Antarmuka

Berdasarkan uji coba yang telah dilakukan untuk menguji fungsionalitas aplikasi antarmuka sistem, berikut ini hasil dari uji coba yang dilakukan :

5.3.3.1 Hasil Uji Coba Membaca Port Serial

Menurut skenario pada Tabel 5.5, yang pertama kali dilakukan adalah menghubungkan node serve ke komputer tempat aplikasi antarmuka berjalan menggunakan kabel USB. Setelah itu aplikasi akan mencari port serial yang aktif. Kemudian, aplikasi akan membaca data dari port serial tersebut. Pada uji coba yang telah dilakukan, didapatkan hasil seperti pada Gambar 5.8. Hal ini menunjukkan bahwa fungsionalitas penerimaan paket data dari node *cluster head* berjalan dengan baik.

```
Stable Library
=========================================
Native lib Version = RXTX-2.1-7
Java lib Version   = RXTX-2.1-7
COM Port Found
1,10,4.30,1.00,8.00,99.58
1,10,4.30,1.00,7.00,100.00
BUILD SUCCESSFUL (total time: 15 seconds)
```

Gambar 5.6 Hasil Uji Coba Membaca Port Serial

5.3.3.2 Hasil Uji Coba Menampilkan Data

Menurut skenario pada Tabel 5.6, yang pertama kali dilakukan adalah aplikasi memebaca data dari port serial milik node server. Setelah itu aplikasi akan memantau apakah data dapat ditampilkan atau tidak. Pada uji coba yang telah dilakukan, didapatkan hasil seperti pada Gambar 5.7. Dari gambar tersebut, terdapat hal penting yang perlu diperhatikan. Yaitu, node yang

menjadi *cluster head* memiliki warna *background* yang berbeda dari node yang lain.

Gambar 5.7 Hasil Uji Coba Menampilkan Data

5.3.3.3 Hasil Uji Coba Memasukkan Data ke Database

Menurut skenario pada Tabel 5.7, yang pertama kali dilakukan adalah aplikasi membaca data dari port serial. Setelah itu aplikasi akan memeriksa apakah data error atau tidak. Kemudian, aplikasi akan menjalankan fungsi untuk memasukkan data ke database. Pada uji coba yang telah dilakukan, didapatkan hasil seperti pada Gambar 5.8. Hal ini menunjukkan bahwa fungsionalitas memasukkan data ke database berjalan dengan baik.

ch_node	jumlah	rata_rata	datamin	datamax	waktu
1	10	6.1	4	9	2017-06-01 14:07:34
1	10	4.2	1	8	2017-06-01 14:07:40
2	10	4.4	1	9	2017-06-01 14:10:40
2	10	4.3	2	7	2017-06-01 14:10:46
2	10	4.8	3	8	2017-06-01 14:10:50
3	10	5.5	1	9	2017-06-01 14:10:56
3	10	6	4	9	2017-06-01 14:11:00
3	10	5.7	2	9	2017-06-01 14:11:06
1	10	6.9	2	9	2017-06-01 14:11:10
1	10	4.7	1	9	2017-06-01 14:11:16
1	10	4.9	1	8	2017-06-01 14:11:20

Gambar 5.8 Hasil Uji Coba Memasukkan Data ke Database

5.4 Uji Coba Performa

Pada jaringan ini, ada beberapa hal penting yang performanya harus diuji. Uji coba performa pada sistem ini dilakukan dengan mengecek keberhasilan pengiriman data ke server dan mengukur penurunan daya tahan baterai dengan membandingkan sistem dengan jaringan yang tidak menggunakan metode pergantian *cluster head* dengan mengubah *delay* pengiriman dan jumlah data yang diolah.

Uji coba performa ini dilakukan di sebuah lapangan yang berlokasi di perumahan kompleks Blok U Kampus ITS. Gambar 5.9 menunjukkan tampilan lapangan beserta lokasi peletakan node.

Gambar 5.9 Lapangan Lokasi Uji Coba Performa

Pada gambar diatas, dapat dilihat lokasi uji coba performa yang akan dilakukan. Nomor-nomor yang berada pada Gambar 5.9 menunjukkan lokasi peletakkan node-node klien dalam jaringan. Nomor 1 menunjukkan lokasi node 1, nomor 2 menunjukkan lokasi node 2, dan node 3 menunjukkan lokasi node 3. Gambar 5.10, Gambar 5.11, dan Gambar 5.12 menunjukkan lebih jelas peletakkan node-node dalam jaringan.

Gambar 5.10 Peletakkan node 1

Gambar 5.11 Peletakkan node 2

Gambar 5.12 Peletakkan node 3

Berikut di bawah ini pengujian – pengujian yang dilakukan.

5.4.1 Uji Coba dengan *Delay* Pengiriman Sebesar 0.5 Detik dan Data yang Diolah Sebanyak 30 Data

Pada uji coba ini, masing-masing node klien akan mengirimkan data ke node *cluster head* dengan interval waktu setiap 0.5 detik. Kemudian node *cluster head* akan mengumpulkan 30 data sebelum melakukan pengolahan data dan mengirim hasilnya ke node server. Pada uji coba ini, ada 3 hal yang akan diuji. Yang pertama adalah tingkat keberhasilan pengiriman data, yang kedua adalah penurunan daya baterai, dan yang ketiga adalah perbandingan penggunaan daya dengan jaringan tanpa *cluster head*.

5.4.2 Uji Coba dengan *Delay* Pengiriman Sebesar 1 Detik dan Data yang Diolah Sebanyak 15 Data

Pada uji coba ini, masing-masing node klien akan mengirimkan data ke node *cluster head* dengan interval waktu setiap 1 detik. Kemudian node *cluster head* akan mengumpulkan 15 data sebelum melakukan pengolahan data dan mengirim

hasilnya ke node server. Pada uji coba ini, ada 3 hal yang akan diuji. Yang pertama adalah tingkat keberhasilan pengiriman data, yang kedua adalah penurunan daya baterai, dan yang ketiga adalah perbandingan penggunaan daya dengan jaringan tanpa *cluster head*.

5.4.3 Uji Coba dengan *Delay* Pengiriman Sebesar 2 Detik dan Data yang Diolah Sebanyak 30 Data

Pada uji coba ini, masing-masing node klien akan mengirimkan data ke node *cluster head* dengan interval waktu setiap 2 detik. Kemudian node *cluster head* akan mengumpulkan 30 data sebelum melakukan pengolahan data dan mengirim hasilnya ke node server. Pada uji coba ini, ada 3 hal yang akan diuji. Yang pertama adalah tingkat keberhasilan pengiriman data, yang kedua adalah penurunan daya baterai, dan yang ketiga adalah perbandingan penggunaan daya dengan jaringan tanpa *cluster head*.

5.5 Hasil dan Evaluasi Uji Coba Performa

Pada bagian ini, akan dijelaskan mengenai hasil dari masing-masing uji coba performa yang telah dilakukan. Skenario uji coba yang telah dijabarkan terdiri dari beberapa hal. Secara garis besar, pengujian terhadap perfoma terbagi menjadi 3 yaitu uji coba pengiriman data, uji coba penurunan daya baterai dan uji coba perbandingan dengan jaringan tanpa *cluster head*. Ketiga uji coba tersebut dijalankan pada 3 skenario dengan *delay* pengiriman dan jumlah data olahan yang berbeda.

5.5.1 Hasil Uji Coba dengan *Delay* Pengiriman Sebesar 0.5 Detik dan Data yang Diolah Sebanyak 30 Data

Pada uji coba ini, jaringan dengan *cluster head* dijalankan selama ±5 menit dengan masing-masing node memiliki *delay* waktu pengiriman sebesar 0.5 detik sebelum mengirim data ke

node *cluster head* dan node *cluster head* melakukan pengolahan ketika terdapat 30 data sebelum mengirimkan hasilnya ke node server. Seperti yang dijelaskan sebelumnya, terdapat 3 hal yang diuji.

Pertama adalah uji coba tingkat keberhasilan pengiriman data dari node *cluster head* ke node server. Hasil uji coba dapat dilihat di Tabel 5.8.

Tabel 5.8 Hasil Uji Coba pengiriman dengan *Delay* 0.5 Detik dan Pengolahan 30 Data

	percobaan ke					Total	Persentase
	1	2	3	4	5		
Sukses	61	59	62	57	60	299	92.86%
Gagal	3	4	1	10	5	23	7.14%

Berdasarkan hasil uji coba yang ada pada tabel di atas, ditunjukkan dari total 322 pengiriman paket data, terdapat 23 data yang gagal, sehingga menunjukkan tingkat keberhasilan dari pengiriman data sebesar 92.86%.

Berikutnya adalah uji coba penurunan daya baterai yang terjadi pada masing-masing node klien dalam jaringan. Hasil uji coba dapat dilihat di Tabel 5.9.

Tabel 5.9 Hasil Uji Coba penurunan baterai dengan *Delay* 0.5 Detik dan Pengolahan 30 Data

	Node1	Node2	Node3
Sebelum	100	100	100
Sesudah	63.4	63.79	63.09
Penurunan	36.6	36.21	36.91

Dan kondisi baterai setelah uji coba dapat dilihat di Gambar 5.13, Gambar 5.14, dan Gambar 5.15.

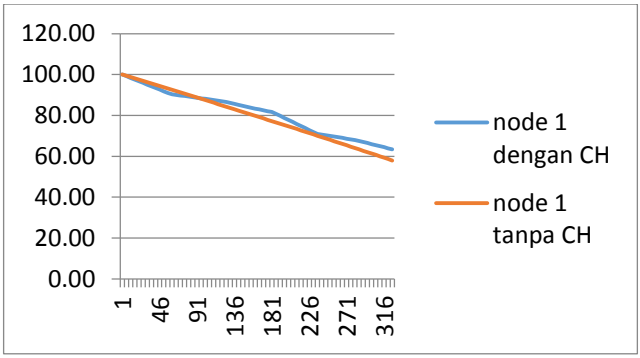
Gambar 5.13 Kondisi baterai node 1

Gambar 5.14 Kondisi baterai node 2

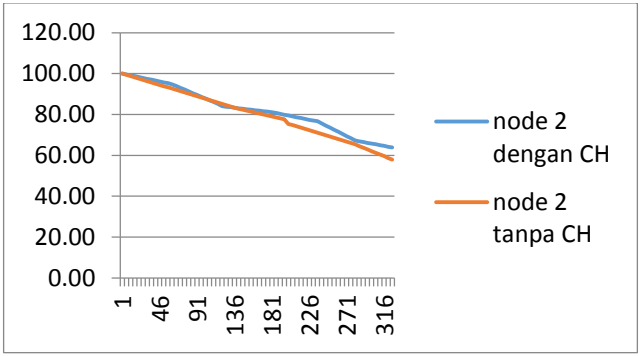
Gambar 5.15 Kondisi baterai node 3

Berdasarkan hasil uji coba yang ada pada tabel di atas, ditunjukkan penurunan daya baterai dari masing-masing node dalam jaringan sebesar ±36%.

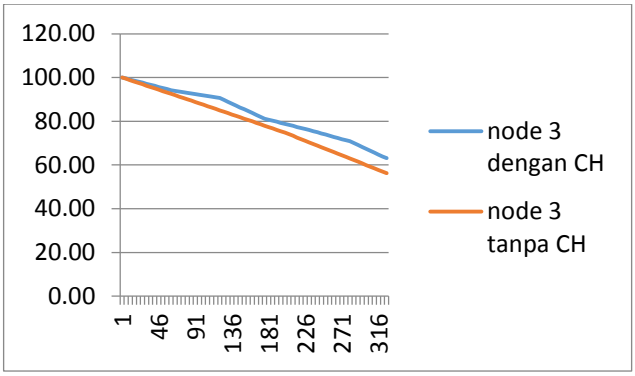
Percobaan yang ketiga adalah uji coba untuk membandingkan performa masing-masing node dengan membandingkan jaingan yang menggunakan *cluster head* dan jaringan yang tidak menggunakan *cluster head.* Hasil uji coba dapat dilihat di Gambar 5.16, Gambar 5.17, dan Gambar 5.18.



Gambar 5.16 Perbandingan baterai node 1



Gambar 5.17 Perbandingan baterai node 2



Gambar 5.18 Perbandingan baterai node 3

Berdasarkan hasil uji coba yang ada pada gambar di atas, ditunjukkan penurunan daya baterai jaringan dalam tugas akhir ini lebih sedikit dibandingkan dengan jaringan yang tidak menggunakan *cluste head*.

5.5.2 Hasil Uji Coba dengan *Delay* Pengiriman Sebesar 1 Detik dan Data yang Diolah Sebanyak 15 Data

Pada uji coba ini, jaringan dengan *cluster head* dijalankan selama ±5 menit dengan masing-masing node memiliki *delay* waktu pengiriman sebesar 1 detik sebelum mengirim data ke node *cluster head* dan node *cluster head* melakukan pengolahan ketika terdapat 15 data sebelum mengirimkan hasilnya ke node server. Seperti yang dijelaskan sebelumnya, terdapat 3 hal yang diuji.

Pertama adalah uji coba tingkat keberhasilan pengiriman data dari node *cluster head* ke node server. Hasil uji coba dapat dilihat di Tabel 5.10.

Tabel 5.10 Hasil Uji Coba dengan *Delay* 1 Detik dan Pengolahan 15 Data

	percobaan ke					Total	Persentase
	1	2	3	4	5		
Sukses	59	58	61	57	63	298	94.01%
Gagal	3	6	2	5	3	19	5.99%

Berdasarkan hasil uji coba yang ada pada tabel di atas, ditunjukkan dari total 298 pengiriman paket data, terdapat 19 data yang gagal, sehingga menunjukkan tingkat keberhasilan dari pengiriman data sebesar 94.01%.

Berikutnya adalah uji coba penurunan daya baterai yang terjadi pada masing-masing node klien dalam jaringan. Hasil uji coba dapat dilihat di Tabel 5.11.

Tabel 5.11 Hasil Uji Coba penurunan baterai dengan *Delay* 1 Detik dan Pengolahan 15 Data

	Node1	Node2	Node3
Sebelum	100	100	100
Sesudah	71.21	70.96	71.62
Penurunan	28.79	29.04	28.38

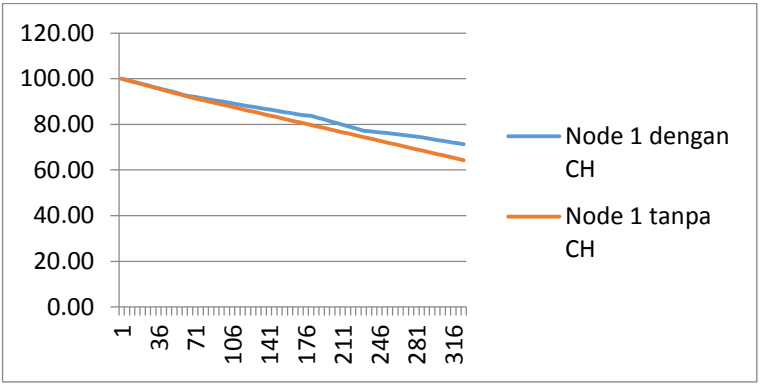
Berdasarkan hasil uji coba yang ada pada tabel di atas, ditunjukkan penurunan daya baterai dari masing-masing node dalam jaringan sebesar ±28%. Dan kondisi baterai setelah uji coba dapat dilihat di Gambar 5.19, Gambar 5.20, dan Gambar 5.21.

Gambar 5.19 Kondisi Baterai Node Pertama

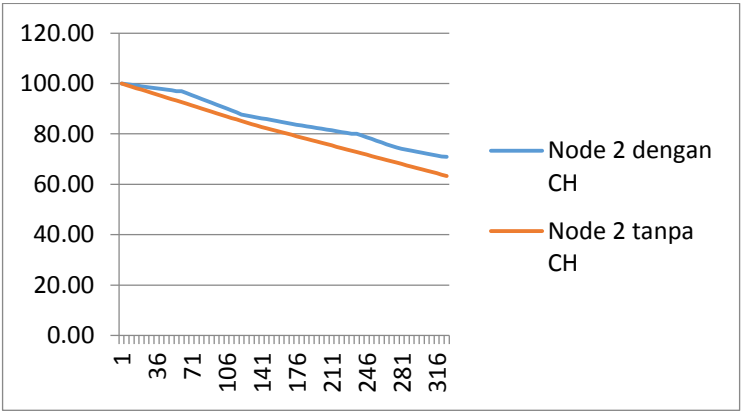
Gambar 5.20 Kondisi Baterai Node Kedua

Gambar 5.21 Kondisi Baterai Node Ketiga

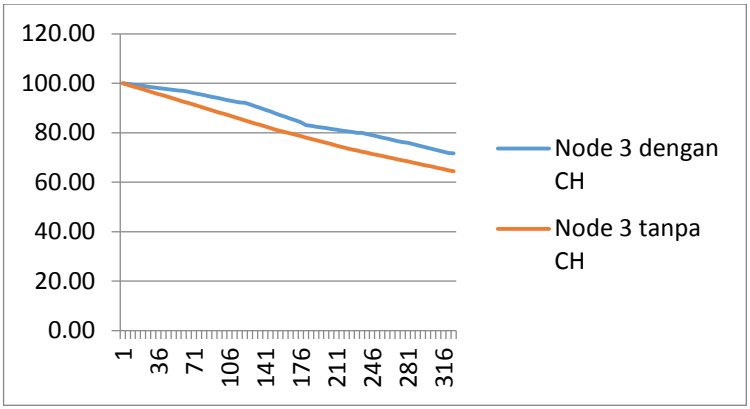
Percobaan yang ketiga adalah uji coba untuk membandingkan performa masing-masing node dengan membandingkan jaingan yang menggunakan *cluster head* dan jaringan yang tidak menggunakan *cluster head.* Hasil uji coba dapat dilihat di Gambar 5.22, Gambar 5.23, dan Gambar 5.24.



Gambar 5.22 Perbandingan Baterai Node Pertama



Gambar 5.23 Perbandingan Baterai Node Kedua



Gambar 5.24 Perbandingan Baterai Node Ketiga

Berdasarkan hasil uji coba yang ada pada gambar di atas, ditunjukkan penurunan daya baterai jaringan dalam tugas akhir ini lebih sedikit dibandingkan dengan jaringan yang tidak menggunakan *cluste head*.

5.5.3 Hasil Uji Coba dengan *Delay* Pengiriman Sebesar 2 Detik dan Data yang Diolah Sebanyak 30 Data

Pada uji coba ini, jaringan dengan *cluster head* dijalankan selama ±5 menit dengan masing-masing node memiliki *delay* waktu pengiriman sebesar 2 detik sebelum mengirim data ke node *cluster head* dan node *cluster head* melakukan pengolahan ketika terdapat 30 data sebelum mengirimkan hasilnya ke node server. Seperti yang dijelaskan sebelumnya, terdapat 3 hal yang diuji.

Pertama adalah uji coba tingkat keberhasilan pengiriman data dari node *cluster head* ke node server. Hasil uji coba dapat dilihat di Tabel 5.12.

Tabel 5.12 Hasil Uji Coba dengan *Delay* 2 Detik dan Pengolahan 30 Data

	percobaan ke					Total	persentase
	1	2	3	4	5		
sukses	17	15	16	18	18	84	91.30%
gagal	5	7	3	0	0	8	8.70%

Berdasarkan hasil uji coba yang ada pada tabel di atas, ditunjukkan dari total 84 pengiriman paket data, terdapat 8 data yang gagal, sehingga menunjukkan akurasi dari pengiriman data sebesar 91.30%.

Berikutnya adalah uji coba penurunan daya baterai yang terjadi pada masing-masing node klien dalam jaringan. Hasil uji coba dapat dilihat di Tabel 5.13.

Tabel 5.13 Hasil Uji Coba penurunan baterai dengan *Delay* 2 Detik dan Pengolahan 30 Data

	Node1	Node2	Node3
Sebelum	100	100	100
Sesudah	64.91	65.04	64.33
Penurunan	35.09	34.96	35.67

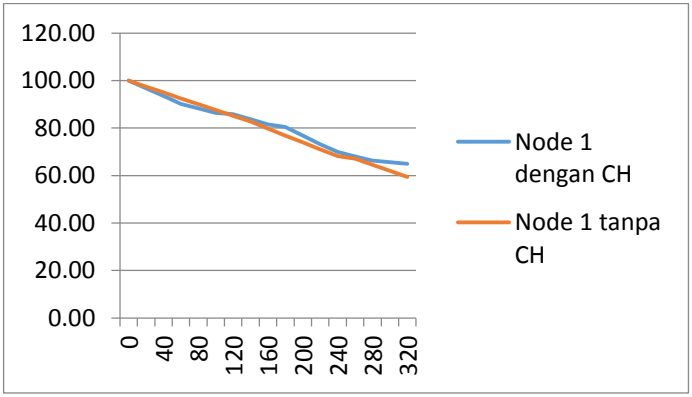
Berdasarkan hasil uji coba yang ada pada tabel di atas, ditunjukkan penurunan daya baterai dari masing-masing node dalam jaringan sebesar ±35%. Dan kondisi baterai setelah uji coba dapat dilihat di Gambar 5.25, Gambar 5.26, dan Gambar 5.27.

Gambar 5.25 Status Baterai Node 1

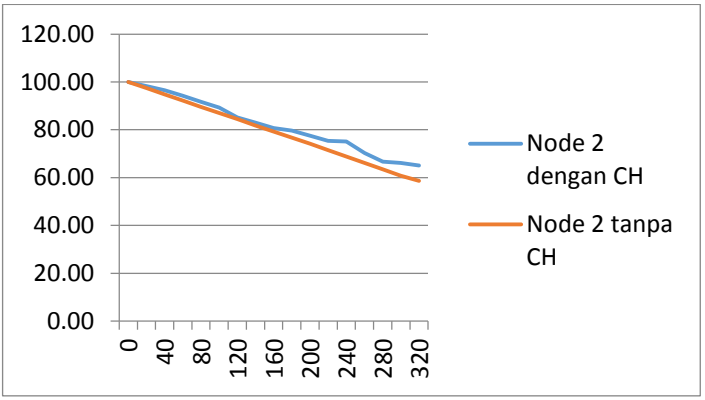
Gambar 5.26 Status Baterai Node 2

Gambar 5.27 Status Baterai Node 3

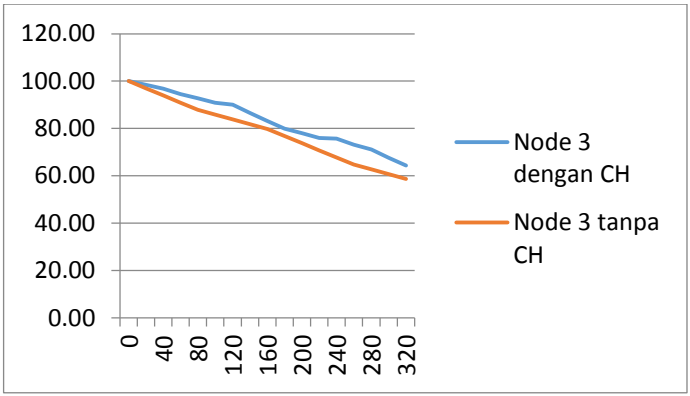
Percobaan yang ketiga adalah uji coba untuk membandingkan performa masing-masing node dengan membandingkan jaingan yang menggunakan *cluster head* dan jaringan yang tidak menggunakan *cluster head.* Hasil uji coba dapat dilihat di Gambar 5.28, Gambar 5.29, dan Gambar 5.30.



Gambar 5.28 Perbedaan Baterai Node 1



Gambar 5.29 Perbedaan Baterai Node 2



Gambar 5.30 Perbedaan Baterai Node 3

Berdasarkan hasil uji coba yang ada pada gambar di atas, ditunjukkan penurunan daya baterai jaringan dalam tugas akhir ini lebih sedikit dibandingkan dengan jaringan yang tidak menggunakan *cluste head*.

(Halaman ini sengaja dikosongkan)

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari tujuan pembuatan sistem dan hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, dijabarkan pula saran yang diharapkan dapat berguna untuk pengembangan sistem dan penelitian selanjutnya.

6.1 Kesimpulan

Kesimpulan yang didapatkan berdasarkan hasil uji coba fungsional dan performa adalah sebagai berikut :

1. Jaringan Sensor Nirkabel yang dibangun menggunakan modul *wireless* nRF24L01+ dapat melakukan pergantian *cluster head* secara dinamis secara terus menerus dengan menerapkan metode *round-robin*.
2. Node *cluster head* dapat melakukan pengolahan data sesuai dengan gelombang data yang ingin diolah. Hasil olahan berupa nilai rata-rata, minimal dan maksimal.
3. Node *cluster head* dapat memilih node *cluster head* selanjutnya dengan menggunakan metode *round-robin*.
4. Aplikasi antarmuka sistem dapat menampilkan simulasi pergantian *cluster head*, menampilkan data hasil olahan dan dapat memasukkan data yang diambil ke dalam database.
5. Jaringan yang menggunakan *cluster head* penggunaan baterainya lebih sedikit dibandingkan jaringan yang tanpa menggunakan *cluster head*.
6. Tingkat keberhasilan pengiriman yaitu sebesar ±90%.

6.2 Saran

Saran yang diberikan untuk pengembangan penelitian kedepannya adalah sebagai berikut :

1. Mengembangkan metode pemilihan *cluster head* yang lain agar performa jaringan lebih meningkat.
2. Disarankan untuk mencoba dengan jumlah node yang lebih banyak.
3. Pengujian kedepannya sebaiknya juga menggunakan berbagai macam perangkat tambahan sebagai sumber datanya.

DAFTAR PUSTAKA

[1] "Wireless Sensor Node With the NRF24L01," *Instructables.com*. [Daring]. Tersedia pada: http://www.instructables.com/id/Wireless-Sensor-Node-With-the-NRF24L01/. [Diakses: 02-Jun-2017].

[2] G. Kannan dan T. Sree Renga Raja, "Energy efficient distributed cluster head scheduling scheme for two tiered wireless sensor network," *Egypt. Inform. J.*, vol. 16, no. 2, hal. 167–174, Jul 2015.

[3] P. Azad dan V. Sharma, "Cluster Head Selection in Wireless Sensor Networks under Fuzzy Environment," *Int. Sch. Res. Not.*, vol. 2013, hal. e909086, Feb 2013.

[4] R. H. Arpaci-Dusseau dan A. C. Arpaci-Dusseau, *Operating Systems: Three Easy Pieces [Chapter: Scheduling Introduction]*. Arpaci-Dusseau Books, 2014.

[5] A. Kadir, *Buku Pintar Pemrograman Arduino*. Yogyakarta: Mediakom, 2014.

[6] eZ Systems, "nRF24L01 / 2.4GHz RF / Products / Home - Ultra Low Power Wireless Solutions from NORDIC SEMICONDUCTOR." [Daring]. Tersedia pada: http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01. [Diakses: 02-Jun-2017].

[7] "MySQL :: MySQL 5.7 Reference Manual :: 1.3.1 What is MySQL?" [Daring]. Tersedia pada: https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html. [Diakses: 02-Jun-2017].

[8] J. Gosling, B. Joy, G. L. Steele, G. Bracha, dan A. Buckley, *The Java Language Specification, Java SE 8 Edition*, 1st ed. Addison-Wesley Professional, 2014.

[9] T. B. Woehr Jesse Glick, Simeon Greene, Vaughn Spurlin, Jack J., *NetBeans: The Definitive Guide*. .

[10] I. P. E. Pratama dan S. Suakanto, *Wireless Sensor Network*. Bandung: Informatika, 2015.

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Moch. Imam Zarqoni merupakan anak dari pasangan Bapak Munthohar Amin (Almarhum) dan Ibu Siti Fatwiyatun. Lahir di Bojonegoro pada tanggal 30 Januari 1995. Penulis menempuh pendidikan formal dimulai dari RA Darul Ulum (RA DU) Pasinan (1998-2000), Madrasah Ibtidaiyah Darul Ulum Pasinan (2000-2006), MTsN Model Babat (2006-209), SMK Telkom Malang (2009-2012) dan sesudah lulus dari SMK Telkom Malang melanjutkan menimba ilmu di jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember, Surabaya (2013-2017). Bidang studi yang diambil oleh penulis pada saat berkuliah di Teknik Informatika ITS adalah Komputasi Berbasis Jaringan (KBJ). Penulis aktif dalam organisasi seperti KMI (2015-2016) sebagai Staf Ahli Departemen Syiar Keluarga Muslim Informatika 2015-2016. Penulis memiliki hobi bermain game dan membaca novel. Penulis dapat dihubungi melalui email: imamzarqoni95@gmail.com.